(12)

AD_____

COMPUTER MODELING AND OPTIMIZATION OF OBOGS WITH CONTAMINANTS

ANNUAL REPORT

JOSEPH J. BEAMAN

OCTOBER 10, 1986

Supported by

U.S. ARMY MEDICAL RESEARCH AND DEVELOPMENT COMMAND
Fort Detrick, Frederick, Maryland 21701-5012

Contract No. DAMD17-84-C-4076

University of Texas at Austin
Department of Mechanical Engineering
Austin, TX 78712-1063

Approved for public release; distribution unlimited

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

SECURITY CLASSIFICATION OF THIS PAGE

AD-A176 037

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704 0188
Exp Date Jun 30 1986

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION The University of Texas at Austin	6b OFFICE SYMBOL (If applicable)
6c ADDRESS (City, State, and ZIP Code) Department of Mechanical Engineering Austin, Texas 78712	7a NAME OF MONITORING ORGANIZATION 7b ADDRESS (City, State, and ZIP Code)
8a NAME OF FUNDING / SPONSORING ORGANIZATION U.S. Army Medical Researcl. & Development Command	8b OFFICE SYMBOL (If applicable) SGRD-RMI-S
8c ADDRESS (City, State, and ZIP Code) Fort Detrick Frederick, Maryland 21701-5012	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DAMD17-84-C-4076

10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO 3E1-	TASK NO	WORK UNIT ACCESSION NO
62777A	62777A878	AF	154

11 TITLE (Include Security Classification)

(U) Computer Modeling and Optimization of OBOGS with Contaminants

12 PERSONAL AUTHOR(S)
Beaman, Joseph J.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Annual	FROM 7/15/85 TO 7/14/86	1986 October 10	54

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	
06	11		Aeromedical; Oxygen concentrators; Molecular sieve;
09	02		Computer modeling; Contaminants

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This report covers the second year of a project dealing with the development of an OBOGS computer model to aid in design and specification of these oxygen generation systems for the Army's in-flight, medivac, and field hospital use. A simple three component model using uncoupled linear isotherms has been developed but not yet tested due to a lack of data. Also, the two component model for oxygen-nitrogen has been modified to include a power law correlation for the parameters in the isotherm expressions. While this correlation is entirely empirical, it improves performance of the model at low temperatures. A major change in the project has been the transfer of the experimental duties from USAARL to UT. A preliminary set of isotherm experiments has been done (using available equipment at Brooks Air Force Base) and equipment has been ordered for the experiments to be done here at UT. The model is currently being rewritten and generalized for three components with general isotherms using a new equilibrium model.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION	
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED	
22a NAME OF RESPONSIBLE INDIVIDUAL Mary Frances Bostian	22b TELEPHONE (Include Area Code) 301/663-7325	22c OFFICE SYMBOL SGRD-RMI-S

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

TABLE OF CONTENTS

STATEMENT OF PROBLEM

The ultimate goal of this research is to insure proper design of molecular sieve oxygen generation systems for the U. S. Army's in-flight, medivac, and field hospital use. Specifically the research involves further development of an OBOGS model to include the effects of contaminants in the feed air. This OBOGS model can be used to optimize and design OBOGS systems with respect to system parameters such as cycle time and bed and valve dimensions.

BACKGROUND

The pressure swing adsorption process (PSA) and its utility for oxygen generation in military applications were discussed in the last annual report (dated September 13, 1986). PSA can also be used to generate nitrogen (using zeolite 4A in the packed beds), which is useful for fuel tank inerting. See the last annual report for a discussion of the PSA process.

A major development since the last report has been the transfer of the experimental work from USAARL to UT. Thus, much effort has been expended to develop experimental techniques that will allow the determination of unknown model parameters.

APPROACH TO THE PROBLEM

Chromatographic experiments will be done to estimate model parameters. These experiments should yield information on both equilibrium and diffusion phenomena. A mass spectrometer has been ordered and should be delivered at the end of this month. This device will potentially allow for determination of ternary equilibria, where the use of isotopically labeled components may be necessary. Once the experimental arrangement has been tested against known data, work can begin on contaminants with unknown adsorptive properties. Experimental apparatus have been designed for both single column and dual column experiments.

A new equilibrium model will be implemented to allow for the use of more general isotherms in the overall model. This will eliminate the assumptions implicit in the coupling scheme presently used. The new model can predict multicomponent isotherms from pure component data. A variation of this model contains terms for "tuning" the isotherms generated if adequate experimental data exist.

In addition, the mass balance equations will probably be modified to include axial mass dispersion. Dispersion may be important at the zero velocity point in the bed.

A new computational scheme will be required to solve the new system of equations. Adaptive grid finite element methods are being considered.

RESULTS AND CONCLUSIONS

Progress has been made in two areas: experimental and computer modeling. The experimental work has just begun, while the computer work is a continuation and expansion of previous efforts.

The present computer model performs well for the oxygen-nitrogen system. This good performance is the result, to some extent, of the fact that these gases are very similar in their physical properties. The system is equilibrium dominated, so diffusional effects are minimized. Similar molecular weights and diameters further "simplify" the calculational load. Recent modifications fixed some compiler-specific "bugs" (only discovered when the model was compiled from source code at a remote location) and included a power law correlation for the isotherm parameters with temperature (see Appendix A.) (1). This temperature correlation extends the range of applicability of the model. However, these recent modifications do not address the problems that can be expected as the model is applied to more complex systems. The model is being rewritten to include multi-component feed mixtures, new equilibrium expressions and multiple mass transfer coefficients.

The number of components in the feed stream is presently fixed at two (specifically, oxygen-nitrogen). This is to be expanded to three. The properties of the third component (contaminant) are likely to differ greatly from those of oxygen and nitrogen. Therefore, simplifications possible in the oxygen-nitrogen system (such as simple isotherm forms and mass transfer expression) will not apply to the ternary system.

In the present model, oxygen is assumed to have a linear isotherm while nitrogen has a simple Langmuir-type isotherm (2). The isotherm parameters are calculated from an analysis that is specific to these isotherm forms. The isotherms are coupled using the concept of the Ideal Adsorbed Solution (3). The Ideal Adsorbed Solution Theory (IAST) is valid only for similar molecules, and is computationally intractible for ternary solutions.

As a preliminary step to a general ternary model, a limited ternary model has been developed (see Appendix B.). In its present form, this model uses uncoupled linear isotherms with a common mass transfer coefficient. The isotherm slopes are not correlated with temperature. This model represents a limiting case as the simplest possible ternary model. Unfortunately, it has not been tested due to a lack of experimental data.

The general expansion to three components presents problems in two equilibrium areas: the form of the pure component isotherm for the third component and the method of coupling the isotherms. Because the contaminant will most likely be present at low concentrations, a linear isotherm might be used. However, the contaminant will be probably be quite different from oxygen and/or nitrogen so IAST will not apply. Also, some highly polar contaminants may exhibit non-linear behavior even at low concentrations due to localized interactions in the zeolite.

A literature search has provided a model that appears to be able to solve the pure component isotherm and coupling problems. The Vacancy Solution Model (VSM) (4,5,6,7) includes terms for both adsorbate-adsorbate and adsorbate-adsorbent non-ideality. The VSM can be correlated with temperature and can include information from available binary isotherms to improve the ternary prediction. Calculation of activity coefficients may be time consuming, however, in the VSM. A sensitivity study may be done to determine if these need to be calculated at every time/distance step.

A preliminary code for the basic VSM (4,5) has been written, but not yet fully tested.

In the present model, the dynamics of mass transfer are represented in the classical form: constant mass transfer coefficient with linear driving force. The coefficient is the same for both oxygen and nitrogen; its value is determined empirically. For the oxygen-nitrogen system, performance is relatively insensitive to the value of the mass transfer coefficient because the system is equilibrium dominated (diffusion is rapid for both components). The system equations are simplified because of the common mass transfer coefficient.

Larger contaminant molecules will not diffuse as rapidly as oxygen and/or nitrogen. Therefore, at the very least, the third component will need to be considered separately. The addition of this equation may make the model stiff and more difficult to calculate. This equation will require a value for the mass transfer coefficient for the third component. At present, a literature search is underway to determine the best approach to predicting this coefficient, as well as the form of the equation. It is most likely that the present form will be retained with a separate expression for each component. However, models with the mass transfer resistance lumped at the zeolite crystal surface and bidisperse models (with resistances at both the pellet and crystal surfaces) are also being studied.

The inclusion of axial dispersion, along with the changes mentioned above, will make a new numerical scheme necessary. Finite element methods are being considered. Because

of the near "shock" phenomena expected in the concentration profiles through the beds, a very fine grid may be required for adequate resolution. Unfortunately, fine grids are very time intensive to calculate. Two adaptive grid methods are under review. One uses preset "levels" of adaptation to cause fine grids to be used where necessary; more coarse grids are used elsewhere. This method appears to suffer from the fact that nodes are being created and lost, which may complicate its implementation. The other method uses a constant number of nodes; the nodes are "moved" to form a tight grid where it is required. This method suffers from problems associated with the equations of motion for the nodes.

The main structure of the program will be changed to provide more modularity. This modularity is important in order to allow for flexibility in designing an OBOGS with the simulator. As three bed systems and systems with stratified beds (with layers of different zeolite types) become common, the model will be required to simulate them. In its present form, this will necessitate a major rewrite of the code for each change. A more modular approach will make model growth more convenient and efficient. Specifically, if temperature effects are to be addressed (i.e., a non-isothermal bed) in a future implementation of the program, it will be a great benefit if the bed equation section can be easily removed and replaced instead of rewriting the entire code.

The first structural change will be the movement of bed structure functions from the simulator program to a utility program. The designed bed systems will be stored as data files for use by the simulator. This will remove much "overhead" from the simulator as well as reduce the number of times the program needs to be recompiled when making bed parameter changes. A graphics library (for the DEC Regis VT terminals) as well as a possible structure for utility data files have been completed.

Isotherm information for many pure substances on zeolites is not readily available. Diffusion information is even more rare. Air separation data on zeolites 5A, 4A, and 13X are available in the literature because the process is important and common. However, information on other substances is either completely lacking, at inappropriate conditions (for instance, in catalytic cracking processes which operate at elevated temperatures and pressures), or "proprietary."

Therefore, experimental work must be done to determine values for necessary model parameters. Chromatographic techniques hold out the promise of obtaining both isotherm and diffusion information relatively quickly and simply (8,9). Isotherm parameters can be obtained by measuring pulse retention volumes at low flowrates. Diffusion data can be

obtained from a moments analysis on the outlet peak at higher flowrates. The moments analysis usually assumes a symmetric peak (for a pulse experiment) and linear isotherms, however; its applicability to the OBOGS problem will be unknown until some preliminary experiments have been done.

Most of the chromatographic approaches to determination of isotherm parameters usually depend on an assumption of the linear isotherm for the components at a given composition. For small enough concentration pulses, this may be a reasonable assumption. The assumption of a linear isotherm for a concentration pulse is best when an isotope of one of the components is used as the pulse (9). The pulse can then be large (because the bulk concenetration will not change, no matter what the pulse size) and, consequently, more easily detected. The slope of the isotherms at each composition (which are the data obtained from the chromatographic analysis) must be integrated to obtain the isotherm itself. Then, the isotherm parameters can be estimated from the isotherm (for the VSM model (without temperature correlation) there will be three parameters to estimate). The fitting procedure used will have a critical effect on the quality of the parameters estimated. Traditional least squares analysis generally assumes all of the error is in the dependent variable. In our work, however, the independent variable (partial pressure) is also measured and subject to error. A recent paper (10) has noted that while the least squares method will fit the pure component isotherms well, the fact that the error in the independent variable was ignored will cause significant deviations when one attempts to predict binary (or higher) isotherms from these pure component parameters. Therefore, the Error-in-Variables Method (EVM) is being considered for use in parameter estimation (11). This method accounts for error in both measurements through the use of covariance matrices based on many experiments. Construction of these matrices would be quite time consuming. However, it appears as though published covariance matrices based on similar experiments may be applicable to our work (10).

Some preliminary work on the isotherm analysis has been done already, using available equipment at Brooks Air Force Base. These experiments were done in an attempt to duplicate previous work (8) in order to validate the method. Due to equipment limitations, only the binary oxygen-nitrogen isotherm was determined. Unfortunately, during the analysis of these experiments, errors were found in the mathematical derivations in (8) which led to non-physical results. The equations were solved again and physically realizable isotherms were obtained (the corrected equations are in Appendix C.). However, as the analysis in (8) was faulty, it was decided not to compare our results with those in the paper. A comparison to volumetric data on this mixture will be done (it would be better to

compare to data on the mixture at constant total pressure, but binary data of this type is rare) to check on the validity of the method.

Single and double bed apparatus have been designed for the anticipated experiments. A mass spectrometer has been ordered and is expected to arrive at the end of this month. Experimental work can then begin here at UT.

REFERENCES

1. D. Walshak, Master's Thesis, University of Texas at Austin, to be published.

2. J. J. Beaman, A. J. Healey, and J. Werlin, Trans. ASME J. Dynamic Sys., Meas., and Control, 105(1983) 265.

3. A. L. Myers and J. M. Prausnitz, AIChE J., 11(1965) 121.

4. S. Suwanayuen and R. P. Danner, AIChE J., 26(1980) 68.

5. S. Suwanayuen and R. P. Danner, AIChE J., 26(1980) 76.

6. T. W. Cochran, R. L. Kabel, and R. P. Danner, AIChE J., 31(1985) 268.

7. T. W. Cochran, R. L. Kabel, and R. P. Danner, AIChE J., 31(1985) 2075.

8. E. van der Vlist and J van der Meijden, J. Chromatography, 79(1973) 1.

9. J. J. Haydel and R. Kobayashi, Ind. Eng. Chem Fund., 6(1967) 546.

10. M. S. High and R. P. Danner, AIChE J., 32(1986) 1138.

11. P. M. Reilly and H. Patino-Leal, Technometrics, 23(1981) 221.

APPENDICES

A. Isotherm Temperature Correlation Equations

B. Simple Ternary Feed OBOGS Model

C. Corrected Equations from van der Vlist

APPENDIX A.

The temperature correlation of the isotherm parameters was entirely empirical work done by Walshak (1). Subroutines CAP1, INITIAL, and MENU need to be modified to implement the temperature correlation; they are included in this appendix. The temperature correlations are:

For oxygen:

$$K_A = .1423275 + 1.83744e\text{-}3 * (T) + 7.2443e\text{-}6 * (T)^2$$

For nitrogen:

$$K_B = .03654573 + 9.5775e\text{-}4 * (T) + 1.321e\text{-}5 * (T)^2$$

$$B = 3.059125$$

T is in degrees Celsius

```
      SUBROUTINE CAP1
      REAL KA,KB
      COMMON/CAP/KA,KB,B,D,TEMP1
      COMMON/VOID/E
      KA=.1423275+.00183744*TEMP1+.0000072443*TEMP1**2
      KB=.03654573+.00095775*TEMP1+.00001321*TEMP1**2
      B=3.0591250816
      WRITE(7,550)TEMP1,B,D,E,KA,KB
      WRITE(8,550)TEMP1,B,D,E,KA,KB
  550 FORMAT('    THE CURRENT BED PARAMETERS ARE:',/,
     1' 12.',T5,' TEMPERATURE = ',T30,F8.3,T40,'DEGREES C',/,
     2' 13.',T5,' B = ',T30,F8.4,/,
     3' 14.',T5,' DIFFUSION COEFFICIENT = ',T30,F8.2,T40,'1/SEC',/,
     4' 15.',T5,' VOID FRACTION = ',T30,F8.3,/,
     5 T5,' AT THIS TEMPERATURE, KA = ',T30,F8.4,T40,
     6 'KGMOLES O2 ADSORBED/KGMOLE GAS',/,
     7 T5,21X,' KB = ',T30,F8.4,T40,'KGMOLES N2 ADSORBED/KGMOLE GAS')
       RETURN
       END
```

```
      SUBROUTINE INITIAL(C1,C2,N1,N2,X,P)
      REAL L,MA,MB,NAO1,NAO2,KA,KB,N1,N2,NBO1,NBO2,LIN
      DIMENSION N1(300,2),N2(300,2),X(3),P(2)
      COMMON/BEDV/LUMPS,DZ,L,D1,D2,MA,MB,RT,AI,AO,AMWO1,AMWL1,AMWO2
     1 ,AMWL2,WBR
      COMMON/CAP/KA,KB,B,D,TEMP1
      COMMON/VOID/E
      COMMON/GEOM/DBYIN,DVSIN,DVOIN,LIN,DIAOIN,DIAIIN
C*************************************************************
C
C              RT = GAS CONSTANT*TEMP (N-M/KGMOLE)
C     AITOT,AOTOT = INLET AND OUTLET CONCENTRIC BED AREAS (M**2)
C           MA,MB = KGMOLECULAR WEIGHTS (KG/KGMOLE)
C             O,L = SUBSCRIPTS FOR Z=O AND Z=L
C             AMW = AVERAGE MOLECULAR WEIGHT (KG/KGMOLE)
C             1,2 = SUBSCRIPTS FOR BED 1 AND BED 2
C               E = VOID FRACTION
C           TEMP1 = TEMPERATURE (DEGREES C)
C*************************************************************
      RT=8314.*(TEMP1+273.16)
      AITOT=3.14159*(DIAIIN*.0254)**2/4.
      AOTOT=3.14159*(DIAOIN**2-DIAIIN**2)*.0254**2/4.
      L=.0254*LIN
      PRINT *,' BED LENGTH (M) = ',L
      PRINT *,' INLET BED AREA (M**2) = ',AITOT
      PRINT *,' OUTLET BED AREA (M**2) = ',AOTOT
      MA=32.
      PRINT *,' VOID FRACTION = ',E
      MB=28.
      AI=AITOT*E
      AO=AOTOT*E
      AMWO1=MA*X(1)+MB*(1.-X(1))
      AMWL1=MA*X(2)+MB*(1.-X(2))
      AMWO2=AMWL1
      AMWL2=MA*X(3)+MB*(1.-X(3))
      C1=P(1)*6.895E3/RT
      C2=P(2)*6.895E3/RT
      CA1=C1*X(1)
      CA2=C2*X(3)
      CB1=C1-CA1
      CB2=C2-CA2
      NBO1=(CB1/KB)/(1.+CB1/(KB*B))
      NAO1=CA1/KA
      NBO2=(CB2/KB)/(1.+CB2/(KB*B))
      NAO2=CA2/KA
      DO 1 I=1,LUMPS
      N1(I,1)=NAO1+NBO1
      N1(I,2)=NAO1
      N2(I,1)=NAO2+NBO2
1     N2(I,2)=NAO2
      DZ=L/FLOAT(LUMPS-1)
      RETURN
      END
```

```
      SUBROUTINE MENU
      REAL LIN,KA,KB,LIN1,KA1,KB1
      DIMENSION TI(11),NY(11)
      COMMON/SYST/DT,PSUP,POUT,TF,WBRL,VS,VO,IIN,TCYC
      COMMON/GEOM/DBYIN,DVSIN,DVOIN,LIN,DIAOIN,DIAIIN
      COMMON/BEDV/LUMPS,DZ,L,D1,D2,MA,MB,RT,AI,AO,AMWO1,AMWL1,AMWO2,
     .            AMWL2,WBR
      COMMON/VOID/E
      COMMON/CAP/KA,KB,B,D,TEMP1
      COMMON/NFINISH/NEXIT
      COMMON/STEP/NSTEP,TSTEP1,TSTEP2,BRSTEP
      COMMON/NPRINT/NOUT1,TI,NY
      NEXIT=0
      NSTEP=0
   10 CALL SYST1
      CALL GEOM1
      CALL CAP1
      WRITE(7,107)
      WRITE(6,107)
  107 FORMAT(' ENTER CORRESPONDING PARAMETER # TO CHANGE PARAMETER',/,
     1' ENTER "0" TO RUN SIMULATION,',
     2' "59" TO STEP THE BREATHING FLOWRATE',/,
     4' ENTER "99" TO EXIT PROGRAM')
      N=0
      READ *,N
      IF(N.EQ. 0) GOTO  600
      IF(N.EQ.59) GOTO  500
      IF(N.EQ.99) GOTO  900
      EPSI=.0000001
      WRITE(7,120)
      WRITE(6,120)
  120 FORMAT(///,'*****************************************************',
     1       //,' ENTER THE PARAMETER AND PRESS "RETURN"',/,
     2//T40,'CURRENT VALUE',/)
      IF(N.LT.6)GOTO 100
      IF(N.LT.12)GOTO 200
      IF(N.LE.18)GOTO 300
      GOTO 800
  100 CONTINUE
      IF(N-2)159,155,155
  155 IF(N-3)164,171,171
  171 IF(N-4)170,175,174
  159 WRITE(6,160) PSUP
      WRITE(7,160) PSUP
  160 FORMAT(' SUPPLY PRESSURE (PSIA)',T40,F8.4,/)
      READ *,PSUP1
      IF(PSUP1.LT.ESPI) GOTO 10
      PSUP=PSUP1
      GOTO 10
  164 WRITE(6,166) POUT
  166 FORMAT(' OUTLET PRESSURE (PSIA)',T40,F8.4,/)
      READ *,POUT1
      IF(POUT1.LT.EPSI) GOTO 10
      POUT=POUT1
      GOTO 10
  170 WRITE(6,172) TF
  172 FORMAT(' FINAL OBSERVATION TIME (SEC)',T40,F8.4,/)
      READ *,TF1
      IF(TF1.LT.EPSI) GOTO 10
      TF=TF1
      GOTO 10
  175 WRITE(6,177) TCYC
  177 FORMAT(' CYCLE TIME (SEC)',T40,F8.4,/)
```

```
      READ *,TCYC1
      IF(TCYC1.LT.EPSI) GOTO 10
      TCYC=TCYC1
      GOTO 10
  174 WRITE(6,176) WBRL
  176 FORMAT(' BREATHING MASS FLOWRATE (STD LIT/MIN)',T40,F8.4,/)
      READ *,WBRL1
      IF(WBRL1.LT.EPSI) GOTO 178
      WBRL=WBRL1
      GOTO 10
  178 GO TO 10
  200 CONTINUE
      IF(N-7) 209,215,205
  205 IF(N-9) 225,235,207
  207 IF(N-10) 235,245,255
  209 WRITE(6,210) DBYIN
  210 FORMAT(' BY-PASS VALVE DIAMETER (IN)',T40,F8.4,/)
      READ *,DBYIN1
      IF(DBYIN1.LT.EPSI) GOTO 10
      DBYIN=DBYIN1
      GO TO 10
  215 WRITE(6,220) DVSIN
  220 FORMAT(' SUPPLY VALVE DIAMETER (IN)',T40,F8.4,/)
      READ *,DVSIN1
      IF(DVSIN1.LT.EPSI) GOTO 10
      DVSIN=DVSIN1
      GOTO 10
  225 WRITE(6,230) DVOIN
  230 FORMAT(' OUTLET VALVE DIAMETER (IN)',T40,F8.4,/)
      READ *,DVOIN1
      IF(DVOIN1.LT.EPSI) GOTO 10
      DVOIN=DVOIN1
      GOTO 10
  235 WRITE(6,240) LIN
  240 FORMAT(' BED LENGTH (IN)',T40,F8.4,/)
      READ *,LIN1
      IF(LIN1.LT.EPSI) GOTO 10
      LIN=LIN1
      GOTO 10
  245 WRITE(6,250) DIAOIN
  250 FORMAT(' OUTER BED DIAMETER (IN)',T40,F8.4,/)
      READ *,DIAOIN1
      IF(DIAOIN1.LT.EPSI) GOTO 10
      DIAOIN=DIAOIN1
      GOTO 10
  255 WRITE(6,260) DIAIIN
  260 FORMAT(' INNER BED DIAMETER (IN)',T40,F8.4,/)
      READ *,DIAIIN1
      IF(DIAIIN1.LT.EPSI) GOTO 10
      DIAIIN=DIAIIN1
      GOTO 10
  300 CONTINUE
      IF(N-13)309,325,305
  305 IF(N-15)335,345,10
  309 WRITE(6,310) TEMP1
  310 FORMAT(' TEMPERATURE (C)',T40,F8.4,/)
      READ *,TEMP2
      IF(TEMP2.LT.EPSI) GOTO 10
      TEMP1=TEMP2
      GOTO 10
  325 WRITE(6,330) B
  330 FORMAT(' B',T40,F8.4,/)
      READ *,B1
```

```
      IF(B1.LT.EPSI) GOTO 10
      B=B1
      GOTO 10
  335 WRITE(6,340) D
  340 FORMAT(' DIFFUSION COEFFICIENT (>100.)',T40,F8.4,/)
      READ *,D1
      IF(D1.LT.EPSI) GOTO 10
      D=D1
      GOTO 10
  345 WRITE(6,350) E
  350 FORMAT(' VOID FRACTION (<1.)',T40,F8.4,/)
      READ *,E1
      IF(E1.LT.EPSI) GOTO 10
      E=E1
      GOTO 10
  500 WRITE(6,510) TF
      WRITE(7,510) TF
  510 FORMAT(////'*****************************************************'//,
     1' ENTER THE TIME AT WHICH THE STEP CHANGE',/
     2' IN BREATHING FLOWRATE IS TO OCCUR',//,
     3' THE FINAL OBSERVATION TIME IS CURRENTLY ',F7.2,' SECONDS',///)
      NSTEP=1
      READ *,TSTEP1
  515 WRITE(6,518) TF
      WRITE(7,518) TF
  518 FORMAT(////'*****************************************************'//,
     1' ENTER THE TIME AT WHICH THE STEP CHANGE',/
     2' IN BREATHING FLOWRATE IS TO END',//,
     3' THE FINAL OBSERVATION TIME IS CURRENTLY ',F7.2,' SECONDS',///)
      READ *,TSTEP2
      WRITE(7,520) TSTEP1,TSTEP2,WBRL
      WRITE(6,520) TSTEP1,TSTEP2,WBRL
  520 FORMAT(//'*****************************************************'//,
     1' ENTER THE NEW BREATHING FLOWRATE OCCURING AS',/,
     2' A STEP CHANGE FROM T= ',F7.2,' TO T= ',F7.2,' SECONDS',//,
     3' THE CURRENT BREATHING FLOWRATE IS',F6.2,'  STD LIT/MIN',///)
      READ *,BRSTEP
      WRITE(7,530) BRSTEP,TSTEP1,TSTEP2
      WRITE(6,530) BRSTEP,TSTEP1,TSTEP2
  530 FORMAT(////,
     1' *****************************************************',////,
     2' THE BREATHING FLOWRATE OF ',F6.2,'  STD LIT/MIN',/
     3' WILL BE INPUT AS A STEP AT T= ',F7.2, '  SECONDS'/,
     3' AND WILL END AT T= ',F7.2, ' SECONDS'////,
     4' *****************************************************'////)
      GOTO 10
  600 CONTINUE
      WRITE(7,610)
      WRITE(6,610)
  610 FORMAT(//////,'*******************************',/////,
     .' THE FOLLOWING PLOTS ARE AVAILABLE TO YOU ON THE TERMINAL',///,
     .' 1   MOLE FRACTION OF OXYGEN VS.TIME (DATA FILE=MOLEFRAC.DAT)',/,
     .' 2   INLET MASS FLOWRATE VS. TIME (DATA FILE=FLOWRATE.DAT)',/,
     .' 3   OUTLET MASS FLOWRATE VS. TIME (DATA FILE=FLOWRATE.DAT)',/,
     .' 4   A DYNAMIC SIMULATION FOR ONE CYCLE (OXYGEN VS. TIME,',/,
     .'     DATA FILE=PROFILE.DAT)',
     .////,
     .' PRESS THE CORRESPONDING # TO HAVE THE OUTPUT SHOWN AS THE',/,
     .' SIMULATION TRANSPIRES',/////)
      READ(5,630)NOUT1
  630 FORMAT(I1)
      IF(NOUT1.EQ.4) GOTO 750
      GO TO 400
```

```
750   DTI=1.0
      DO J=1,10
      TI(J)=TI(J-1)+DTI
      NY(J)=IFIX(TI(J)/DT)
      END DO
      TI(11)=TCYC
      NY(11)=IFIX(TI(11)/DT)
      GO TO 400
 400  WRITE(6,410)
      WRITE(7,410)
 410  FORMAT(////,' YOU HAVE THE FOLLOWING OPTIONS : ',
     ./////,T10,'1. ENTER "0" TO RUN SIMULATION.',/,
     .T10,'2. ENTER "19" TO CHANGE THE PARAMETERS.',/,
     .T10,'3. ENTER "99" TO EXIT PROGRAM.',/////,
     .' ENTER THE CORRESPONDING # AND PRESS "RETURN"',/////)
      READ *, N69
      IF(N69.EQ.0) GOTO  820
      IF(N69.EQ.19) GOTO 10
      IF(N69.EQ.99) GOTO 900
 800  CONTINUE
      WRITE(7,810)
      WRITE(6,810)
 810  FORMAT(/////,'*****************************',/////,
     1' THE PARAMETER VALUE WAS NOT INPUT CORRECTLY',/
     2' TRY AGAIN',/////,'*****************************',////)
      GOTO 10
 820  WRITE(6,825)
      WRITE(7,825)
 825  FORMAT(////,' DO YOU WISH TO DOUBLE THE NUMBER OF SPACE LUMPS IN'/,
     1' THE SIMULATION FOR BETTER ACCURACY ? ( CURRENT NUMBER USED '/,
     2' IS 50 )  (Y/N)',/////)
      READ (5,830) KEYIN
 830  FORMAT ( A )
      IF (KEYIN.EQ.YES) THEN
      LUMPS = 101
      ELSE
      LUMPS = 51
      END IF
      GO TO 1000
 900  CONTINUE
      NEXIT=1
1000  RETURN
      END
```

APPENDIX B.

This appendix contains the simple ternary feed OBOGS model computer code. The model uses simple linear uncoupled isotherms for all components with a common mass transfer coefficient.

```
      PROGRAM MSEV
C
C     OBOGS WITH CONTAMINANT
C     A = COMPONENT A (OXYGEN)
C     B = COMPONENT B (CONTAMINANT)
C     C = COMPONENT C (NITROGEN)
C
      REAL LIN,KA,KB,KC
      REAL L50
      BYTE CONTINUE
      BYTE LOWC
      BYTE KEYIN
      BYTE NULL
      BYTE YES
      BYTE NO
      CHARACTER*64 TEMP
      DIMENSION W(3),XA(3),XB(3),CVA(3),PS(2),P(2)
      DIMENSION WV(502,2),TV(500),XV(502,3)
      DIMENSION DL(500),CA1(500),CA2(500),CB1(500),CB2(500),
     1          XA1(502,11),XA2(502,11),XB1(502,11),XB2(502,11)
      DIMENSION IXABS1(501,11),IYABS1(501,11),IXABS2(501,11),
     1          IYABS2(501,11)
      DIMENSION IX1(501),IY1(501),IX2(501),IY2(501)
      DIMENSION JX(6,11),JY(6,11),JXERSH(6,11),JYERSH(6,11),
     1          JXERSL(6,11),JYERSL(6,11)
      DIMENSION TI(6), NY(6)
      DIMENSION XCA1(502,11),XCA2(502,11),XCB1(502,11),
     1          XCB2(502,11),XBR02(500)
      dimension range(4),xaxis(10),yaxis(10),range2(4),xaxis2(10),
     1          yaxis2(10),xaxis3(10),yaxis3(10),range3(4),
     1          xaxis50(10),yaxis50(10),range50(4)
      DIMENSION L50(402),T50(400),ERR(400)
      COMMON/TIME/T
      COMMON/SYST/DT,PSUP,POUT,TF,WBRL,VS,V0,IIN,TCYC
      COMMON/GEOM/DBYIN,DVSIN,DVOIN,LIN,DIAOIN,DIAIIN
      COMMON/CAP/KA,KB,KC,BB,BC,D
      COMMON/VOID/E
      COMMON/NFINISH/NEXIT
      COMMON/STEP/NSTEP,TSTEP1,TSTEP2,BRSTEP
      COMMON/NPRINT/NOUT1,TI,NY
      COMMON/BEDV/LUMPS,DZ
      COMMON/HILOBED/NBED,NB
      DATA CONTINUE/67/
      DATA LOWC/99/
      DATA NULL/0/
      DATA YES/89/
      DATA NO/78/
17    CALL MENU
      IF ( LUMPS .EQ. 101 ) THEN
      IINC = 3
      ELSE
      IINC = 6
      END IF
      IF ( NEXIT .EQ. 1 ) GOTO 900
      IF ( NSTEP .EQ. 1 ) GOTO 20
      TSTEP1 = 10000000.
      TSTEP2 = 10000000.
      BRSTEP = WBRL
   20 CONTINUE
      DLIN = LIN / (LUMPS-1)
      DO I = 1,LUMPS
      DL(I) = (I-1) * DLIN
      END DO
```

```
      OPEN (UNIT=1, FILE='MOLEFRAC',STATUS='NEW')
      OPEN (UNIT=2, FILE='FLOWRATE',STATUS='NEW')
      OPEN (UNIT=3, STATUS='SCRATCH')
      OPEN (UNIT=4, FILE='PROFILE',STATUS='NEW')
      OPEN (UNIT=8, STATUS='SCRATCH')
      OPEN (UNIT=9, FILE='REGIS',STATUS='NEW')
      OPEN (UNIT=10,FILE='CONTA',STATUS='NEW')
      WRITE(6,7)
    7 FORMAT(//,
     1    '*********************************************')
      PRINT *
      PRINT *,'  OBOGS SIMULATION'
      PRINT *
      IIN = 0
      ND = 0
      N50 = 0
      CDBY = .056 / DBYIN
      IF ( CDBY .GT. 1. ) CDBY = 1.
      IF ( CDBY .LT. .6 ) CDBY = .6
      DBYM = DBYIN * .0254
      PRINT *,' BYPASS VALVE DISCHARGE COEFFICIENT = ',CDBY
      BYVA = 3.14159*DBYM**2 / 4.
      VS = .8 * ( .0254*DVSIN )**2 * 3.14159 / 4.
      V0 = .8 * ( .0254*DV0IN )**2 * 3.14159 / 4.
      PRINT *,' AREA OF BY-PASS (M**2) = ',BYVA
      PRINT *,' CD*AREA SUPPLY VALVE (M**2) = ',VS
      PRINT *,' CD*AREA OUTLET VALVE (M**2) = ',V0
      PRINT *,' SUPPLY PRESSURE(PSI)=',PSUP,
     .        'OUTLET PRESSURE(PSI)=',POUT
      CVA(2) = CDBY*BYVA
      T = 0.0
      DT = .01
      DATAP = 200.
      INC = IFIX ( TF / (DATAP*DT) )
      NT = 0
      IM = 0
      P(1) = 14.5
      P(2) = 14.5
      XA(2) = .20
      XB(2) = 0.
      PRINT *,' BREATHING FLOW (STD LIT/MIN) = ',WBRL
      XABR = .20
      XBBR = 0.
      IF(NOUT1-1)35,35,36
   35 CLOSE(UNIT=2,STATUS='DELETE')
      CLOSE(UNIT=4,STATUS='DELETE')
      CLOSE(UNIT=9,STATUS='DELETE')
      CLOSE(UNIT=10,STATUS='DELETE')
      WRITE(1,57)
   57 FORMAT('  OXYGEN MOLE FRACTION',24X,'TIME')
      GO TO 1
   36 IF(NOUT1.EQ.4) GOTO 37
      IF(NOUT1.EQ.5) GOTO 170
      CLOSE(UNIT=1,STATUS='DELETE')
      CLOSE(UNIT=4,STATUS='DELETE')
      CLOSE(UNIT=9,STATUS='DELETE')
      CLOSE(UNIT=10,STATUS='DELETE')
      WRITE(2,58)
   58 FORMAT('  INLET MASS FLOWRATE',T25,' OUTLET MASS FLOWRATE',
     1' T55,TIME')
      GO TO 1
  170 CLOSE(UNIT=1,STATUS='DELETE')
      CLOSE(UNIT=2,STATUS='DELETE')
```

```
      CLOSE(UNIT=4,STATUS='DELETE')
      CALL FRAME1
      GO TO 1
37    CLOSE(UNIT=1,STATUS='DELETE')
      CLOSE(UNIT=2,STATUS='DELETE')
      CLOSE(UNIT=10,STATUS='DELETE')
      CALL FRAME1
1     CONTINUE
      XA(1) = .21
      XA(3) = .21
      XB(1) = .21
      XB(3) = .21
      CALL TIMEF ( CVA,PH,PS,T )
      CALL BEDS ( P, W, XA, XB, XABR, XBBR, CVA, PS, DT,
     .            CA1, CA2, CB1, CB2, C1, C2 )
      IF ( NOUT1 .EQ. 4 .OR. NOUT1 .EQ. 5 ) GO TO 77
      GO TO 78
77    DO 1000 K=1,8
      IF ( NT .EQ. NY(K) ) GO TO 150
      GO TO 1000
150   CONTINUE
      IF ( K .GT. 1 ) THEN

      CALL INITIAL1 ( IY1(1), IY2(1), NB )
      DO J=2,LUMPS
      DLIN = LIN / (LUMPS-1)
      DL(J) = (J-1) * DLIN
      IF ( NB .EQ. 0 ) THEN
      CALL DRAWF ( IX1(J), IY1(J), 1 )
      ELSE
      CALL DRAWF ( IX2(J), IY2(J), 1 )
      END IF
      END DO
      CALL INITIAL2 ( IY1(1), IY2(1), NB )
      DO M=2,LUMPS
      IF ( NB .EQ. 0 ) THEN
      CALL DRAWB ( IX2(M), IY2(M), 1 )
      ELSE
      CALL DRAWB ( IX1(M), IY1(M), 1 )
      END IF
      END DO
      ELSE
      END IF

      DO J=2,LUMPS
      DLIN = LIN / (LUMPS-1)
      DL(J) = (J-1) * DLIN
      END DO
      IF ( TTI .GT. 0.0 ) THEN
      CALL NEWTIME ( TTI,1 )
      CALL NEWTIME ( TI(K),0 )
      IF ( NBED .EQ. NB ) THEN
      CONTINUE
      ELSE
      CALL FRAME2 ( NB,1 )
      CALL FRAME2 ( NBED,0 )
      END IF
      ELSE
      CALL NEWTIME ( TI(K),0 )
      CALL FRAME2 ( NBED,0 )
      END IF
      TTI = TI(K)
      IF (NOUT1 .EQ. 4) GO TO 120
```

```
        WRITE(10,*) PH,NBED,T,NT,NY(K)
      WRITE(10,110) TI(K)
      GO TO 130
 120  WRITE(4,110) TI(K)
 110  FORMAT('  BED COORDINATE',T20,'MOLE FRACTION IN BED#1',
     1T46,'MOLE FRACTION IN BED#2',3X,'(TIME T=',F8.2,'SEC.)',/)
130   CONTINUE
      DO L=1,LUMPS
      XCA1(L,K) = CA1(L)
      XCA2(L,K) = CA2(L)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      XCB1(L,K) = CB1(L)
      XCB2(L,K) = CB2(L)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      IF ( NBED .EQ. 0 ) THEN
      XA1(L,K) = XCA1(L,K) / C1
      XA2(L,K) = XCA2(L,K) / C2
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      XB1(L,K) = XCB1(L,K) / C1
      XB2(L,K) = XCB2(L,K) / C2
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      IXABS1(L,K) = 275 + IINC*(L-1)
      IXABS2(L,K) = 575 - IINC*(L-1)
      IF ( NOUT1 .EQ. 4 ) THEN
        CALL SCALE1 ( XA1(L,K), IYABS1(L,K) )
        CALL SCALE2 ( XA2(L,K), IYABS2(L,K) )
       ELSE
        CALL SCALE1 ( XB1(L,K), IYABS1(L,K) )
        CALL SCALE2 ( XB2(L,K), IYABS2(L,K) )
      END IF
      ELSE
      M = LUMPS+1-L
      XA1(M,K) = XCA1(L,K) / C1
      XA2(M,K) = XCA2(L,K) / C2

      XB1(M,K) = XCB1(L,K) / C1
      XB2(M,K) = XCB2(L,K) / C2

      IXABS2(L,K) = 275 + IINC*(L-1)
      IXABS1(L,K) = 575 - IINC*(L-1)
      IF ( NOUT1 .EQ. 4 ) THEN
        CALL SCALE1 ( XA1(M,K), IYABS1(M,K) )
        CALL SCALE2 ( XA2(M,K), IYABS2(M,K) )
       ELSE
        CALL SCALE1 ( XB1(M,K), IYABS1(M,K) )
        CALL SCALE2 ( XB2(M,K), IYABS2(M,K) )
      END IF
      END IF
      END DO

      CALL INITIAL1 ( IYABS1(1,K), IYABS2(1,K), NBED )
      DO I=2,LUMPS
      IF ( NBED .EQ. 0 ) THEN
      CALL DRAWF ( IXABS1(I,K), IYABS1(I,K), 0 )
      IX1(I) = IXABS1(I,K)
      IY1(I) = IYABS1(I,K)
      ELSE
      CALL DRAWF ( IXABS2(I,K), IYABS2(I,K), 0 )
      IX2(I) = IXABS2(I,K)
      IY2(I) = IYABS2(I,K)
      END IF
      END DO
      DO I=1,LUMPS
```

```
      IF (NOUT1 .EQ. 4) GO TO 146
      GO TO 140
146   WRITE(4,300) DL(I), XA1(I,K), XA2(I,K)
      GO TO 160
140   WRITE(10,300)DL(I), XB1(I,K), XB2(I,K)
300   FORMAT ( 5X, F6.3, 14X, F10.6, 16X, F10.6 )
160   CONTINUE
      END DO

      CALL INITIAL2 ( IYABS1(1,K), IYABS2(1,K), NBED )
      DO 255 N=2,LUMPS
      IF ( NBED .EQ. 0 ) THEN
      CALL DRAWB ( IXABS2(N,K), IYABS2(N,K), 0 )
      IX2(N) = IXABS2(N,K)
      IY2(N) = IYABS2(N,K)
      ELSE
      CALL DRAWB ( IXABS1(N,K), IYABS1(N,K), 0 )
      IX1(N) = IXABS1(N,K)
      IY1(N) = IYABS1(N,K)
      END IF
      NB = NBED
255   CONTINUE

      XBR022 = XABR / 1.05
      DY = XBR022
      DPY = DY * 280.0
      IDY = IFIX ( DPY )
      IABS = 412 - IDY
      IF ( K .GT. 1 ) THEN
      WRITE(9,500) IA
500   FORMAT ( ' P[695,'I3']' )
      IF(IABS.LT.IA) THEN
      ID = IA - IABS
      WRITE(9,510) ID
510   FORMAT ( ' W(R,NO)V[]V[695,-'I3']' )
      ELSE
      ID = IABS - IA
      WRITE(9,520) ID
520   FORMAT ( ' W(R,N1)V[]V[695,+'I3']' )
      END IF
      ELSE
      WRITE(9,530) IABS
530   FORMAT ( ' P[695,412]V[,'I3']' )
      END IF
      WRITE(9,535)
535   FORMAT ( ' W(NO)' )
C      WRITE(6,200)DL(I),XA1(I,K),XA2(I,K)
C  200 FORMAT(8X,F6.3,11X,E10.4,7X,E10.4)

      range(1) = 0.
      range(2) = 18.
      range(3) = 0.
      range(4) = 1.
      nxaxis = 20
      encode (nxaxis,258,xaxis)
258   format ( 'BED COORDINATE (IN.)' )
      nyaxis = 13
      encode (nyaxis,369,yaxis)
369   format ( 'MOLE FRACTION' )
      nxaxis50 = 11
      encode (nxaxis50,256,xaxis50)
256   format ( 'TIME (SEC.)' )
      nyaxis50 = 9
```

```
      encode (nyaxis50,366,yaxis50)
366   format ( 'L50 (IN.)' )
      IA = IABS
 1000 CONTINUE
      IF(((NOUT1.EQ.5).AND.(MOD(NT,INC).EQ.0)).OR.((NOUT1.EQ.5)
     .    .AND.(AMOD(T,TCYC).EQ.0.))) THEN
      N50 = N50 + 1
      DO M=1,LUMPS
      IF ( NBED .EQ. 0 ) THEN
      ERR(M)=(CB1(M)/C1)-.10
      ELSE
      L = LUMPS+1-M
      ERR(M) = (CB2(L)/C2) - .10
      END IF
      IF ( ABS(ERR(M)) .LE. 0.015 ) THEN
      L50(N50) = DL(M)
      ELSE
      END IF
      END DO
      T50(N50) = T
        WRITE(11,*) T50(N50),L50(N50),NBED,LUMPS
      ELSE
      END IF
      IF ( NT .EQ. NY(6) ) THEN
        RANGE50(1) = 0.
        RANGE50(2) = 0.
        RANGE50(3) = 0.
        RANGE50(4) = 0.
           CALL ZETAPLT(11,1,N50,RANGE50,T50,L50,N50+2,XAXIS50,
     .                  NXAXIS50,,YAXIS50,NYAXIS50)
           call setaplt(10,6,lumps,range,DL,XB1,LUMPS+2,xaxis,
     .                  nxaxis,yaxis,nyaxis)
           call setaplt(20,6,lumps,range,DL,XB2,LUMPS+2,xaxis,
     .                  nxaxis,yaxis,nyaxis)

           CLOSE ( UNIT=9, STATUS='SAVE' )
           REWIND 9
           OPEN ( UNIT=9, FILE='REGIS', STATUS='OLD' )
           CALL TXTERASE (0)
           CALL REGISTART
           CALL PLTERASE
9220       READ ( 9, 9220, END=9999 ) TEMP
           WRITE (3,*) TEMP
           TYPE 9220,TEMP
9220       FORMAT ( A )
           GO TO 9200
9999       CALL REGISOUT
           CLOSE ( UNIT=9, STATUS='DELETE' )
           WRITE(6,9450) NULL
9450       FORMAT ( A, ' PRESS "C" TO CONTINUE : ' )
9400       READ(5,9300) KEYIN
9300       FORMAT ( A )
           IF ( KEYIN .EQ. CONTINUE .OR. KEYIN .EQ. LOWC ) THEN
           CALL TXTERASE (0)
           CALL REGISTART
           CALL PLTERASE
           CALL REGISOUT
           go to 17
           ELSE
           GO TO 9400
           END IF
        ELSE
           GO TO 6
```

```
      ENDIF

78    IF ( MOD(NT,INC) ) 5,5,6
5     IM = IM + 1
      XBR02(IM) = XABR / 1.05
      IF ( NOUT1 .EQ. 1 ) THEN
      WRITE(1,84) XBR02(IM), T
84    FORMAT ( 6X, F10.6, 25X, F10.3)
   82 IF ( ND .EQ. 0 ) THEN
      CALL TXTERASE ( 255 )
      CALL REGISTART
      CALL PLTERASE
      CALL FRAME
      TYPE 80
   80 FORMAT ( '!P[130,330]', $ )
      GO TO 47
      ELSE
      CALL STIME (T, IX, TF, 600., 130 )
      CALL SCALEM ( XBR02(IM), IY, 1.0, 1.0 )
      TYPE 85, IX, IY
   85 FORMAT ( '!V['I3','I3']', $ )
      END IF
      ELSE IF ( NOUT1 .EQ. 2 ) THEN
      IF ( ND .EQ. 0 ) THEN
      CALL TXTERASE ( 255 )
      CALL REGISTART
      CALL PLTERASE
      CALL FRAME
      TYPE 86
   86 FORMAT ( '!P[130,330]', $ )
      GO TO 47
      ELSE
      CALL STIME ( T, IX, TF, 600., 130 )
      CALL SCALEM ( WV(ND,1), IY, 0.08, 0.1 )
      TYPE 87, IX, IY
   87 FORMAT ( '!V['I3','I3']', $ )
      WRITE(2,68) WV(ND,1), WV(ND,2), T
   68 FORMAT ( 8X, F10.6, 11X, F10.6, 10X, F10.3 )
      END IF
      ELSE IF ( NOUT1 .EQ. 3 ) THEN
      IF ( ND .EQ .0 ) THEN
      CALL TXTERASE ( 255 )
      CALL REGISTART
      CALL PLTERASE
      CALL FRAME
      TYPE 88
   88 FORMAT ( '!P[130,330]', $ )
      GO TO 47
      ELSE
      CALL STIME ( T, IX, TF, 600., 130 )
      CALL SCALEM ( WV(ND,2), IY, 0.08, 0.1 )
      TYPE 89, IX, IY
   89 FORMAT ( '!V['I3','I3']', $ )
      WRITE(2,69) WV(ND,1), WV(ND,2), T
   69 FORMAT ( 8X, F10.6, 11X, F10.6, 10X, F10.6, 6X, F10.3 )
      END IF
      ELSE
      GO TO 82
      END IF
   47 CONTINUE
      ND=ND+1
      IF ( PH-180. ) 2,3,3
2     WV(ND,1)=W(3)
```

```
      WV(ND,2)=W(1)
      GO TO 4
3     WV(ND,1)=-W(1)
      WV(ND,2)=-W(3)
4     XV(ND,1)=XBRO2(IM)
      XV(ND,2)=0.
      XV(ND,3)=1.
      TV(ND)=T
6     NT=NT+1
      T = NT * DT

      IF ( T .LT. TSTEP1 .AND. T .Le. TF ) THEN
      WBRLOLD = WBRL
      GO TO 1
      ELSE IF ( T .GE. TSTEP1 .AND. T .LT. TSTEP2 ) THEN
      WBRL = BRSTEP
      GO TO 1
      ELSE IF ( T .GE. TSTEP2 .AND. T .LE. TF ) THEN
      WBRL = WBRLOLD
      GO TO 1
      ELSE
      WBRL = WBRLOLD
      GO TO 18
      END IF

18    CONTINUE
      CALL REGISOUT
      WRITE(6,42) NULL
42    FORMAT ( A, ' PRESS "C" TO CONTINUE : ' )
44    READ(5,41) KEYIN
41    FORMAT ( A )
      IF ( KEYIN .EQ. CONTINUE .OR. KEYIN .EQ. LOWC ) THEN
      CALL TXTERASE (0)
      CALL REGISTART
      GO TO 43
      ELSE
      GO TO 44
      END IF
43    CALL PLTERASE
      CALL REGISOUT
      range2(1) = 0.
      range2(2) = 0.
      range2(3) = 0.
      range2(4) = 1.
      range3(1) = 0.
      range3(2) = 0.
      range3(3) = -.01
      range3(4) = .04
      nxaxis2 = 11
      encode ( nxaxis2, 257, xaxis2 )
257   format ( 'TIME (SEC.)' )
      nyaxis2 = 16
      encode ( nyaxis2, 368, yaxis2 )
368   format ( '02 MOLE FRACTION' )
      nxaxis3 = 11
      encode ( nxaxis3, 852, xaxis3 )
852   format ( 'TIME (SEC.)' )
      nyaxis3 = 18
      encode ( nyaxis3, 963, yaxis3 )
963   format ( 'BREATHING FLOWRATE' )
      call setaplt ( 30, 3, nd, range2, tv, xv, nd+2, xaxis2,
                     nxaxis2, yaxis2, nyaxis2 )
      call setaplt ( 40, 2, nd, range3, tv, wv, nd+2, xaxis3,
```

```
     .                nxaxis3, yaxis3, nyaxis3 )
100   FORMAT ( 14X, 8E12.3 )
      GO TO 17
  900 WRITE(6,910)
  910 FORMAT(///////'*******************************',/////,
     1' PROGRAM TERMINATED BY OPERATOR',
     2//////'*******************************',//////)
      END

      BLOCK DATA
      REAL LIN,KA,KB,KC
      COMMON/SYST/DT,PSUP,POUT,TF,WBRL,VS,V0,IIN,TCYC
      COMMON/GEOM/DBYIN,DVSIN,DVOIN,LIN,DIAOIN,DIAIIN
      COMMON/CAP/KA,KB,KC,BB,BC,D
      COMMON/VOID/E
      DATA DT, PSUP, POUT, TF, WBRL, VS, V0, IIN, TCYC /.01,
     &   40.0, 15.0, 30.0, 10.0, 0.0, 0.0, 0, 10.7 /
      DATA DBYIN, DVSIN, DVOIN, LIN, DIAOIN, DIAIIN /
     &   .075, .306174, .43866, 15.5, 5.73, 2.18 /
      DATA KA, KB, KC, BB, BC, D / .2133, .007597, .05223,
     &   .6414, .6265, 200. /
      DATA E / .37 /
      END

      SUBROUTINE MENU
      REAL LIN,KA,KB,KC,LIN1,KA1,KB1,KC1
      DIMENSION TI(6), NY(6), TI4(6), NY4(6), TI5(6), NY5(6)
      BYTE KEYIN
      BYTE YES
      BYTE loyes
      BYTE NO
      BYTE lono
      COMMON/SYST/DT,PSUP,POUT,TF,WBRL,VS,V0,IIN,TCYC
      COMMON/GEOM/DBYIN,DVSIN,DVOIN,LIN,DIAOIN,DIAIIN
      COMMON/BEDV/LUMPS,DZ,L,D1,D2,MA,MB,MC,RT,AI,A0,AMWO1,
     .            AMWL1,AMWO2,AMWL2,WABR,WBBR
      COMMON/VOID/E
      COMMON/CAP/KA,KB,KC,BB,BC,D
      COMMON/NFINISH/NEXIT
      COMMON/STEP/NSTEP,TSTEP1,TSTEP2,BRSTEP
      COMMON/NPRINT/NOUT1,TI,NY
      DATA YES,loyes/89,121/
      DATA NO,lono/78,110/
      NEXIT=0
      NSTEP=0
   10 CALL SYST1
      CALL GEOM1
      CALL CAP1
      WRITE(7,107)
      WRITE(6,107)
  107 FORMAT(' ENTER CORRESPONDING PARAMETER # TO CHANGE PARAMETER',/,
     1' ENTER "0" TO SIGNIFY PARAMETERS ARE CORRECT--RUN SIMULATION',/
     2' ENTER "50" TO INSERT A STEP CHANGE IN BREATHING FLOWRATE',/,
     4' ENTER "99" TO EXIT PROGRAM')
      N = 0
      READ *,N
      IF ( N .EQ. 0 ) GOTO  600
      IF ( N .EQ. 50 ) GOTO  500
      IF ( N .EQ. 99 ) GOTO  900
      EPSI = .0000001
      WRITE(7,120)
      WRITE(6,120)
  120 FORMAT(///,'*****************************************************',
```

```
     1       //,' ENTER THE PARAMETER AND PRESS "RETURN"',/,
     2//T40,'CURRENT VALUE',/)
      IF ( N .LT. 6 ) GOTO 100
      IF ( N .LT. 12 ) GOTO 200
      IF ( N .LE. 16 ) GOTO 300
      GOTO 800
  100 CONTINUE
      IF ( N-2 ) 159,155,155
  155 IF ( N-3 ) 164,171,171
  171 IF ( N-4 ) 170,175,174
  159 WRITE(6,160) PSUP
      WRITE(7,160) PSUP
  160 FORMAT(' SUPPLY PRESSURE (PSIA)',T40,F8.4,/)
      READ *,PSUP1
      IF(PSUP1.LT.ESPI) GOTO 10
      PSUP=PSUP1
      GOTO 10
  164 WRITE(6,166) POUT
  166 FORMAT(' OUTLET PRESSURE (PSIA)',T40,F8.4,/)
      READ *,POUT1
      IF(POUT1.LT.EPSI) GOTO 10
      POUT=POUT1
      GOTO 10
  170 WRITE(6,172) TF
  172 FORMAT(' FINAL OBSERVATION TIME (SEC)',T40,F8.4,/)
      READ *,TF1
      IF(TF1.LT.EPSI) GOTO 10
      TF=TF1
      GOTO 10
  175 WRITE(6,177) TCYC
  177 FORMAT(' CYCLE TIME (SEC)',T40,F8.4,/)
      READ *,TCYC1
      IF(TCYC1.LT.EPSI) GOTO 10
      TCYC=TCYC1
      GOTO 10
  174 WRITE(6,176) WBRL
  176 FORMAT(' BREATHING MASS FLOWRATE (STD LIT/MIN)',T40,F8.4,/)
      READ *,WBRL1
      IF(WBRL1.LT.EPSI) GOTO 178
      WBRL=WBRL1
      GOTO 10
  178 GO TO 10
  200 CONTINUE
      IF(N-7) 209,215,205
  205 IF(N-9) 225,235,207
  207 IF(N-10) 235,245,255
  209 WRITE(6,210) DBYIN
  210 FORMAT(' BY-PASS VALVE DIAMETER (IN)',T40,F8.4,/)
      READ *,DBYIN1
      IF(DBYIN1.LT.EPSI) GOTO 10
      DBYIN=DBYIN1
      GO TO 10
  215 WRITE(6,220) DVSIN
  220 FORMAT(' SUPPLY VALVE DIAMETER (IN)',T40,F8.4,/)
      READ *,DVSIN1
      IF(DVSIN1.LT.EPSI) GOTO 10
      DVSIN=DVSIN1
      GOTO 10
  225 WRITE(6,230) DVOIN
  230 FORMAT(' OUTLET VALVE DIAMETER (IN)',T40,F8.4,/)
      READ *,DVOIN1
      IF(DVOIN1.LT.EPSI) GOTO 10
      DVOIN=DVOIN1
```

```
      GOTO 10
  235 WRITE(6,240) LIN
  240 FORMAT(' BED LENGTH (IN)',T40,F8.4,/)
      READ *,LIN1
      IF(LIN1.LT.EPSI) GOTO 10
      LIN=LIN1
      GOTO 10
  245 WRITE(6,250) DIAOIN
  250 FORMAT(' OUTER BED DIAMETER (IN)',T40,F8.4,/)
      READ *,DIAOIN1
      IF(DIAOIN1.LT.EPSI) GOTO 10
      DIAOIN=DIAOIN1
      GOTO 10
  255 WRITE(6,260) DIAIIN
  260 FORMAT(' INNER BED DIAMETER (IN)',T40,F8.4,/)
      READ *,DIAIIN1
      IF(DIAIIN1.LT.EPSI) GOTO 10
      DIAIIN=DIAIIN1
      GOTO 10
  300 CONTINUE
      IF(N-13)309,315,305
  305 IF(N-15)325,335,345
  309 WRITE(6,310) KA
  310 FORMAT ( ' KA', T40, F8.4, / )
      READ *, KA1
      IF ( KA1 .LT. EPSI ) GOTO 10
      KA = KA1
      GOTO 10
  315 WRITE(6,320) KB
  320 FORMAT ( ' KB', T40, F8.4, / )
      READ *, KB1
      IF ( KB1 .LT. EPSI ) GOTO 10
      KB = KB1
      GOTO 10
  325 WRITE(6,330) KC
  330 FORMAT ( ' KC', T40, F8.4, / )
      READ *, KC1
      IF( KC1 .LT. EPSI ) GOTO 10
      KC = KC1
      GOTO 10
  335 WRITE(6,340) D
  340 FORMAT(' DIFFUSION COEFFICIENT (>100.)',T40,F8.4,/)
      READ *,D1
      IF(D1.LT.EPSI) GOTO 10
      D=D1
      GOTO 10
  345 WRITE(6,350) E
  350 FORMAT(' VOID FRACTION (<1.)',T40,F8.4,/)
      READ *,E1
      IF(E1.LT.EPSI) GOTO 10
      E=E1
      GOTO 10
  500 WRITE(6,510) TF
      WRITE(7,510) TF
  510 FORMAT(////'*****************************************************'//,
     1' ENTER THE TIME AT WHICH THE STEP CHANGE',/
     2' IN BREATHING FLOWRATE IS TO OCCUR',//,
     3' THE FINAL OBSERVATION TIME IS CURRENTLY ',F7.2,' SECONDS',///)
      NSTEP=1
      READ *,TSTEP1
  515 WRITE(6,518) TF
      WRITE(7,518) TF
  518 FORMAT(////'*****************************************************'//,
```

```
     1' ENTER THE TIME AT WHICH THE STEP CHANGE',/
     2' IN BREATHING FLOWRATE IS TO END',//,
     3' THE FINAL OBSERVATION TIME IS CURRENTLY ',F7.2,' SECONDS',///)
      READ *,TSTEP2
      WRITE(7,520) TSTEP1,TSTEP2,WBRL
      WRITE(6,520) TSTEP1,TSTEP2,WBRL
  520 FORMAT(//'**************************************************'//,
     1' ENTER THE NEW BREATHING FLOWRATE OCCURING AS',/,
     2' A STEP CHANGE FROM T= ',F7.2,' TO T= ',F7.2,' SECONDS',//,
     3' THE CURRENT BREATHING FLOWRATE IS',F6.2,'  STD LIT/MIN',///)
      READ *,BRSTEP
      WRITE(7,530) BRSTEP,TSTEP1,TSTEP2
      WRITE(6,530) BRSTEP,TSTEP1,TSTEP2
  530 FORMAT(////,
     1' **************************************************',////,
     2' THE BREATHING FLOWRATE OF ',F6.2,'  STD LIT/MIN',/
     3' WILL BE INPUT AS A STEP AT T= ',F7.2, '  SECONDS'/,
     3' AND WILL END AT T= ',F7.2, ' SECONDS'////,
     4' **************************************************'////)
      GOTO 10
  600 CONTINUE
      WRITE(7,610)
      WRITE(6,610)
  610 FORMAT(//////,'*****************************',/////,
     .' THE FOLLOWING PLOTS ARE AVAILABLE TO YOU ON THE TERMINAL',///,
     .' 1    MOLE FRACTION OF OXYGEN VS.TIME (DATA FILE=MOLEFRAC.DAT)',/,
     .' 2    INLET MASS FLOWRATE VS. TIME (DATA FILE=FLOWRATE.DAT)',/,
     .' 3    OUTLET MASS FLOWRATE VS. TIME (DATA FILE=FLOWRATE.DAT)',/,
     .' 4    A DYNAMIC SIMULATION FOR ONE CYCLE (OXYGEN VS. TIME,',/,
     .'      DATA FILE=PROFILE.DAT)',/,
     .' 5    DYNAMIC SIMULATION OF PROPAGATION OF CONTAMINANT ALONG',/,
     .'      THE BED LENGTH',/,
     .///,
     .' PRESS THE CORRESPONDING # TO HAVE THE OUTPUT SHOWN AS THE',/,
     .' SIMULATION TRANSPIRES',/////)
      READ(5,630)NOUT1
  630 FORMAT(I1)
      IF (NOUT1 .EQ. 4 ) GOTO 750
      IF (NOUT1 .EQ. 5 ) GOTO 760
      GO TO 400
  750 DO J=1,5
        FJ = FLOAT(J)
        TI4(J) = 2.*FJ - 1.
        NY4(J) = IFIX ( TI4(J)/DT )
      END DO
      TI4(6) = TCYC
      NY4(6) = IFIX ( TI4(6)/DT )
        DO M=1,6
        TI(M) = TI4(M)
        NY(M) = NY4(M)
        END DO
      GO TO 400
  760 DO J=2,5
        FJ = FLOAT(J)
        TI5(J) = ( TF/5. ) * (FJ-1.)
        NY5(J) = IFIX (TI5(J)/DT )
      END DO
      TI5(1) = 1.0
      NY5(1) = IFIX ( TI5(1)/DT )
      TI5(6) = TF
      NY5(6) = IFIX ( TI5(6)/DT )
        DO M=1,6
        TI(M) = TI5(M)
```

```
      NY(M) = NY5(M)
      END DO
400 WRITE(6,410)
    WRITE(7,410)
410 FORMAT(////,' YOU HAVE THE FOLLOWING OPTIONS : ',
   ./////,T10,'1. ENTER "0" TO RUN SIMULATION.',/,
   .T10,'2. ENTER "19" TO CHANGE THE PARAMETERS.',/,
   .T10,'3. ENTER "99" TO EXIT PROGRAM.',/////,
   .' ENTER THE CORRESPONDING # AND PRESS "RETURN"',/////)
    READ *, N69
    IF(N69.EQ.0) GOTO  820
    IF(N69.EQ.19) GOTO 10
    IF(N69.EQ.99) GOTO 900
800 CONTINUE
    WRITE(7,810)
    WRITE(6,810)
810 FORMAT(/////,'*****************************',/////,
   1' THE PARAMETER VALUE WAS NOT INPUT CORRECTLY',/
   2' TRY AGAIN',/////,'*****************************',////)
    GOTO 10
820 WRITE(6,825)
    WRITE(7,825)
825 FORMAT(////,' DO YOU WISH TO DOUBLE THE NUMBER OF SPACE LUMPS IN'/,
   1' THE SIMULATION FOR BETTER ACCURACY ? ( CURRENT NUMBER USED '/,
   2' IS 50 )  (Y/N)',/////)
    READ (5,830) KEYIN
830 FORMAT ( A )
    IF ( (KEYIN .EQ. YES) .OR. (KEYIN .EQ. loyes) ) THEN
    LUMPS = 101
    ELSE
    LUMPS = 51
    END IF
    GO TO 1000
900 CONTINUE
    NEXIT = 1
1000 RETURN
    END

    SUBROUTINE SYST1
    COMMON/SYST/DT,PSUP,POUT,TF,WBRL,VS,V0,IIN,TCYC
    WRITE(6,350) PSUP,POUT,TF,TCYC,WBRL
    WRITE(7,350) PSUP,POUT,TF,TCYC,WBRL
350 FORMAT('   CURRENT SYSTEM PARAMETERS ARE:',/,
   1'  1.',T5,' SUPPLY PRESSURE=',T30,F8.2,T40,'PSIA',/,
   2'  2.',T5,' OUTLET PRESSURE=',T30,F8.2,T40,'PSIA',/,
   3'  3.',T5,' FINAL OBSERVATION TIME=',T30,F8.2,T40,'SEC',/,
   4'  4.',T5,' CYCLE TIME=',T30,F8.2,T40,'SEC',/,
   5'  5.',T5,' BREATHING FLOWRATE=',T30,F8.2,T40,'STD LIT/MIN')
    RETURN
    END

    SUBROUTINE GEOM1
    REAL LIN
    COMMON/GEOM/DBYIN,DVSIN,DVOIN,LIN,DIAOIN,DIAIIN
    WRITE(6,450) DBYIN,DVSIN,DVOIN,LIN,DIAOIN,DIAIIN
    WRITE(7,450) DBYIN,DVSIN,DVOIN,LIN,DIAOIN,DIAIIN
450 FORMAT('   CURRENT GEOMETRIC PARAMETERS ARE:',/,
   1'  6.',T5,' BY-PASS VALVE DIAMETER = ',T30,F8.4,T40,'IN',/,
   2'  7.',T5,' SUPPLY VALVE DIAMETER = ',T30,F8.4,T40,'IN',/,
   3'  8.',T5,' OUTLET VALVE DIAMETER = ',T30,F8.4,T40,'IN',/,
   4'  9.',T5,' BED LENGTH = ',T30,F8.4,T40,'IN',/,
   5' 10.',T5,' OUTER BED DIAMETER = ',T30,F8.4,T40,'IN',/,
   6' 11.',T5,' INNER BED DIAMETER = ',T30,F8.4,T40,'IN')
```

```
      RETURN
      END

      SUBROUTINE CAP1
      REAL KA,KB,KC
      COMMON/CAP/KA,KB,KC,BB,BC,D
      COMMON/VOID/E
      WRITE(7,550)KA,KB,KC,D,E
      WRITE(6,550)KA,KB,KC,D,E
  550 FORMAT('   THE CURRENT BED PARAMETERS ARE:',/,
     1' 12.',T5,' KA = ',T30,F8.4,T40,'KGMOLES O2 ADSORBED/KGMOLE GAS',/,
     2' 13.',T5,' KB = ',T30,F8.4,T40,'KGMOLES CO ADSORBED/KGMOLE GAS',/,
     3' 14.',T5,' KC = ',T30,F8.4,T40,'KGMOLES N2 ADSORBED/KGMOLE GAS',/,
     4' 15.',T5,' DIFFUSION COEFFICIENT = ',T30,F8.2,T40,'1/SEC',/,
     5' 16.',T5,' VOID FRACTION = ',T30,F8.3)
      RETURN
      END

      SUBROUTINE TIMEF(CVA,PH,PS,T)
C------CALCULATES CVA(T) AND PS(T)
      DIMENSION CVA(3),PS(2)
      DATA PHI,PH1,PH2,PH3,PH4,PH5,PH6/10.,20.,160.,180.,
     1 200.,340.,360./
      COMMON/SYST/DT,PSUP,POUT,TF,WBRL,VS,V0,IIN,TCYC
      COMMON/HILOBED/NBED,NB
C*****************************************************************
C
C       PH = PHASE ANGLE OF CYCLE (DEGREES)
C      TCYC = CYCLE TIME (SEC)
C
C*****************************************************************
      I = 0
      PH = AMOD ( 360.*T/TCYC + PHI, 360. )
      IF ( PH-PH1 ) 1,2,2
1     RA = PH / PH1
      GO TO 11
2     IF ( PH-PH2 ) 3,4,4
3     RA = 1.
      GO TO 11
4     IF ( PH-PH3 ) 5,6,6
5     RA = (PH3-PH) / (PH3-PH2)
      GO TO 11
6     IF ( PH-PH4 ) 7,8,8
7     RA = (PH-PH3) / (PH4-PH3)
      I = 1
      GO TO 11
8     IF ( PH-PH5 ) 9,10,10
9     RA = 1.
      I = 1
      GO TO 11
10    RA = (PH6-PH) / (PH6-PH5)
      I = 1
11    NBED = I
      IF ( I ) 12,12,13
12    PS(1) = PSUP
      PS(2) = POUT
      CVA(1) = VS*RA
      CVA(3) = V0*RA
      RETURN
13    PS(1) = POUT
      PS(2) = PSUP
      CVA(1) = V0*RA
      CVA(3) = VS*RA
```

```
      RETURN
      END

      SUBROUTINE INITIAL (C1, C2, N1, N2, XA, XB, P )
      REAL L,MA,MB,MC,NAO1,NAO2,KA,KB,KC,N1,N2,NBO1,NBO2,LIN,NCO1,NCO2
      DIMENSION N1(300,3),N2(300,3),XA(3),XB(3),P(2)
      COMMON/BEDV/LUMPS,DZ,L,D1,D2,MA,MB,MC,RT,AI,AO,AMWO1,AMWL1,AMWO2
     1 ,AMWL2,WBR
      COMMON/CAP/KA,KB,KC,BB,BC,D
      COMMON/VOID/E
      COMMON/GEOM/DBYIN,DVSIN,DVOIN,LIN,DIAOIN,DIAIIN
C-----------------------------------------------------------------
C
C               RT = GAS CONSTANT*TEMP ( N-M / KGMOLE )
C      AITOT,AOTOT = INLET AND OUTLET CONCENTRIC BED AREAS ( M**2 )
C            MA,MB = KGMOLECULAR WEIGHTS ( KG / KGMOLE )
C              O,L = SUBSCRIPTS FOR Z=O AND Z=L
C              AMW = AVERAGE MOLECULAR WEIGHT ( KG / KGMOLE )
C              1,2 = SUBSCRIPTS FOR BED 1 AND BED 2
C                E = VOID FRACTION
C
C-----------------------------------------------------------------
      RT = 8314. * 300.
      AITOT = 3.14159 * (DIAIIN*.0254)**2 / 4.
      AOTOT = 3.14159 * (DIAOIN**2-DIAIIN**2) *.0254**2 / 4.
      L = .0254 * LIN
      PRINT *,' BED LENGTH (M) = ', L
      PRINT *,' INLET BED AREA  (M**2) = ', AITOT
      PRINT *,' OUTLET BED AREA (M**2) = ', AOTOT
      PRINT *,' VOID FRACTION = ', E
      MA = 32.
      MB = 28.
      MC = 28.
      AI = AITOT*E
      AO = AOTOT*E
      AMWO1 = MA*XA(1) + MB*XB(1) + MC*(1.-XA(1)-XB(1))
      AMWL1 = MA*XA(2) + MB*XB(2) + MC*(1.-XA(2)-XB(2))
      AMWO2 = AMWL1
      AMWL2 = MA*XA(3) + MB*XB(3) + MC*(1.-XA(3)-XB(3))
      C1 = P(1)*6.895E3 / RT
      C2 = P(2)*6.895E3 / RT
      CA1 = C1*XA(1)
      CA2 = C2*XA(3)
      CB1 = C1*XB(1)
      CB2 = C2*XB(3)
      CC1 = C1-CA1-CB1
      CC2 = C2-CA2-CB2
      NAO1 = CA1 / KA
      NAO2 = CA2 / KA
      NBO1 = (CB1/KB) / (1.+CB1/(KB*BB))
      NBO2 = (CB2/KB) / (1.+CB2/(KB*BB))
      NCO1 = (CC1/KC) / (1.+CC1/(KC*BC))
      NCO2 = (CC2/KC) / (1.+CC2/(KC*BC))
      DO 1 I=1,LUMPS
      N1(I,1) = NAO1 + NBO1 + NCO1
      N1(I,2) = NAO1
      N1(I,3) = NBO1
      N2(I,1) = NAO2 + NBO2 + NCO2
      N2(I,2) = NAO2
1     N2(I,3) = NBO2
      DZ=L/FLOAT(LUMPS-1)
      RETURN
      END
```

```
      SUBROUTINE BEDS ( P, W, XA, XB, XABR, XBBR, CVA, PS, DTG,
     .                  CA1, CA2, CB1, CB2, C1, C2 )
C
C------SOLVES BED EQUATIONS
C
      REAL L,N1,N2,MA,MB,MC
      DIMENSION CA1(300),CA2(300),CB1(300),CB2(300),N1(300,3),N2(300,3),
     1          G1(300,3),G2(300,3),XAT(2),XBT(2),XA0(2),XB0(2)
     1          ,W(3),XA(3),XB(3),CVA(3),PS(2),PN(2),P(2)
      COMMON/BEDV/LUMPS,DZ,L,D1,D2,MA,MB,MC,RT,AI,A0,AMWO1,AMWL1,AMWO2
     1            ,AMWL2,WBR
      COMMON/VOID/E
      COMMON/SYST/DT,PSUP,POUT,TF,WBRL,VS,V0,IIN,TCYC
      COMMON/CAP/KA,KB,KC,BB,BC,D
      COMMON/TIME/T
C-----------------------------------------------------
C
C         C = TOTAL GAS STREAM CONCENTRATION ( KGMOLE/ M**3 )
C        CA = GAS A STREAM CONCENTRATION ( KGMOLE / M**3 )
C         N = ADSORBED   PHASE CONCENTRATION ( KGMOLE / M**3 )
C         G = GAS PHASE CONCENTRATION ( KGMOLE / M**3 )
C         U = STREAM VELOCITY ( M / SEC )
C         Y = MOLECULAR FLUX ( KGMOLE / M**2-SEC )
C         L = BED   LENGTH    ( M )
C         D = DIFFUSION COEFFICIENT ( 1 / SEC )
C      BETA = D*AREA RATIO    ( 1 / SEC )
C        DT = TIME STEP   ( SEC )
C     LUMPS = NUMBER OF SPACE LUMPS
C       1,2 = SUBSCRIPTS FOR BED 1 AND BED 2
C       VBR = MIXING VOLUME FOR BREATHING FLOW  ( M**3 )
C        BR = SUBSCRIPT INDICATING BREATHING VARIABLE
C        CP = PLENUM MOLE CONCENTRATION    ( KGMOLE / M**3 )
C        VP = BREATHING PLENUM VOLUME   ( M**3 )
C
C-----------------------------------------------------
      DT=DTG
      DO 4 I=1,2
4     PN(I) = PS(I)*6.895E3
      IF(IIN.EQ.1)GO TO 1
      IIN = 1
      CALL INITIAL ( C1, C2, N1, N2, XA, XB, P )
      VPMUL = 1.
      VP = .0018 * VPMUL
      PRINT *,' PLENUM VOLUME (M**3) = ',VP
      DO I=1,7
      PRINT *
      END DO
      WRITE(6,10)NULL
10    FORMAT ( A, ' PLEASE WAIT.........' )
      VBR = .5*VP + .0003
      CP = .95*AMAX1( PN(1),PN(2) ) / RT
      DO 3 I=1,2
      XAT(I) = XABR
3     XBT(I) = XBBR
1     D1 = D
      D2 = D
      WBR = WBRL*4.72E-5 / 2.205
      CALL CONCEN (C1,C2,CA1,CA2,CB1,CB2,W,G1,G2,XA,XB,N1,N2,PN,CVA)
      XABR = XAT(2)
      XBBR = XBT(2)
      AMWBR = MA*XABR + MB*XBBR + MC*(1.-XABR-XBBR)
      DO 5 I=1,2
      XA0(I) = XAT(I)
```

```
5       XB0(I) = XBT(I)
        DO 2 I=1,LUMPS
        N1(I,1) = N1(I,1) - D1*(G1(I,1)-C1)*DT
        N1(I,2) = N1(I,2) - D1*(G1(I,2)-CA1(I))*DT
        N1(I,3) = N1(I,3) - D1*(G1(I,3)-CB1(I))*DT
        N2(I,1) = N2(I,1) - D2*(G2(I,1)-C2)*DT
        N2(I,2) = N2(I,2) - D2*(G2(I,2)-CA2(I))*DT
        N2(I,3) = N2(I,3) - D2*(G2(I,3)-CB2(I))*DT
2       CONTINUE
        FAC = CP * VBR * AMWBR
        TAUI = WBR / FAC
        XAT(1) = XA0(1) + TAUI*(XA(2)-XA0(1))*DT
        XBT(1) = XB0(1) + TAUI*(XB(2)-XB0(1))*DT
        XAT(2) = XA0(2) + TAUI*(XA0(1)-XA0(2))*DT
        XBT(2) = XB0(2) + TAUI*(XB0(1)-XB0(2))*DT
        P(1) = C1*RT / 6.895E3
        P(2) = C2*RT / 6.895E3
        AMWO1 = MA*XA(1) + MB*XB(1) + MC*(1.-XA(1)-XB(1))
        AMWL1 = MA*XA(2) + MB*XB(2) + MC*(1.-XA(2)-XB(2))
        AMWO2 = AMWL1
        AMWL2 = MA*XA(3) + MB*XB(3) + MC*(1.-XA(3)-XB(3))
        RETURN
        END

        SUBROUTINE CONCEN(CG1,CG2,CA1,CA2,CB1,CB2,W,G1,G2,
       1    XA,XB,N1,N2,PN,CVA)
C*****SOLVES FOR CONCENTRATIONS
        REAL L,N1,N2,MA,MB,MC
        DIMENSION CA1(300),CA2(300),CB1(300),CB2(300),N1(300,3),N2(300,3)
       1    ,G1(300,3),G2(300,3)
        DIMENSION GI1(300),GI2(300),PN(2),W(3),XA(3),XB(3),CVA(3)
        COMMON/BEDV/LUMPS,DZ,L,D1,D2,MA,MB,MC,RT,AI,A0,AMWO1,AMWL1,AMWO2
       1    ,AMWL2,WBR
        COMMON/VOID/E
        COMMON/VBEQ/W1,W2,W3,UO1,UL1,UO2,UL2,PN1,PN2,VA1,VA2,VA3
       1    ,BGI1L,BGI1L2,BGI2L,BGI2L2,BL21,BL22,P1,P2
C-------------------------------------------------------
C
C       P = BED PRESSURE    ( N / M**2 )
C      PN = SUPPLY PRESSURES    ( N / M**2 )
C       U = STREAM VELOCITY   ( M / SEC )
C       W = MASS FLOWRATE   ( KG / SEC )
C     CVA = DISCHARGE COEFFICIENT TIMES VALVE AREA ( M**2 )
C       C = TOTAL GAS STREAM CONCENTRATION   ( KGMOLE / M**3 )
C      CA = GAS "A" STREAM CONCENTRATION   ( KGMOLE / M**3 )
C      CB = GAS "B" STREAM CONCENTRATION   ( KGMOLE / M**3 )
C      CC = GAS "C" STREAM CONCENTRATION   ( KGMOLE / M**3 )
C       Z = BED DISTANCE COORDINATE   ( M )
C      YA = MOLECULAR FLUX OF GAS "A" = CA*U   ( KGMOLE / M**2-SEC )
C      YB = MOLECULAR FLUX OF GAS "B" = CB*U   ( KGMOLE / M**2-SEC )
C      XA = MOLE FRACTION OF GAS A
C      XB = MOLE FRACTION OF GAS B
C
C-------------------------------------------------------
        VA1 = CVA(1)
        VA2 = CVA(2)
        VA3 = CVA(3)
        BETA1 = (1.-E)*D1 / E
        BETA2 = (1.-E)*D2 / E
        PN1 = PN(1)
        PN2 = PN(2)
        BL21 = BETA1*.5*L
        BL22 = BETA2*.5*L
```

```
      C1 = CG1
      C2 = CG2
C------CALCULATE G AND INTEGRAL OF G
      GI1(1) = 0.0
      GI2(1) = 0.0
      DO 1 I=1,LUMPS
      CALL CAPZ ( N1, G1, I )
1     CALL CAPZ ( N2, G2, I )
      DO 2 I=2,LUMPS
      IM1 = I-1
      GI1(I) = GI1(IM1) + DZ*( G1(I,1)+G1(IM1,1) ) / 2.
2     GI2(I) = GI2(IM1) + DZ*( G2(I,1)+G2(IM1,1) ) / 2.
      LUM2 = (LUMPS+1) / 2
      BGI1L2 = BETA1 * GI1(LUM2)
      BGI1L = BETA1*GI1(LUMPS) - BGI1L2
      BGI2L2 = BETA2 * GI2(LUM2)
      BGI2L = BETA2*GI2(LUMPS) - BGI2L2
C------SOLVE BED/VALVE EQUATION FOR C
      CALL VAQ ( C1, C2 )
      W(1)=W1
      W(2)=W2
      W(3)=W3
      CG1=C1
      CG2=C2
C------SOLVE FOR CONCENTRATION OF GAS A AND B
      CALL CONCAB ( C1,CA1,CB1,G1,GI1,UO1,UL1,XA(1),XA(2),
     1              XB(1),XB(2),BETA1,AI/A0 )
      CALL CONCAB ( C2,CA2,CB2,G2,GI2,UO2,UL2,XA(2),XA(3),
     1              XB(2),XB(3),BETA2,A0/AI )
      RETURN
      END

      SUBROUTINE CONCAB ( CG, CA, CB, G, GI, UOG, ULG, XAO, XAL,
     1  XBO, XBL, BETAG, ARG )
C------SOLVES FOR CONCENTRATION OF GAS A AND B
      REAL L2,L
      DIMENSION CA(300),CB(300),G(300,3),GI(300),XA(2),XB(2),
     1  U(300),YA(300),YB(300)
      COMMON/BEDV/LUMPS,DZ,L,D1,D2,MA,MB,MC,RT,AI,A0,AMWO1,AMWL1,AMWO2
     1 ,AMWL2,WBR
      COMMON/VOID/E
      COMMON/TIME/T
      C = CG
      BETA = BETAG
      UL = ULG
      UO = UOG
      XA(1) = XAO
      XA(2) = XAL
      XB(1) = XBO
      XB(2) = XBL
      AR = ARG
      LUM2 = (LUMPS+1) / 2
      GIL2 = GI(LUM2)
      L2 = .5*L
      IF ( UO**2 +UL**2 ) 22,22,23

C------ZERO FLOW SOLUTION

22    DO 24 I=1,LUMPS
      CA(I) = G(I,2)
24    CB(I) = G(I,3)
      RETURN
23    CONTINUE
```

```
      CA(1) = C*XA(1)
      CB(1) = C*XB(1)
      U(1) = UO
      YA(1) = UO*CA(1)
      YB(1) = UO*CB(1)
      CA(LUMPS) = C*XA(2)
      CB(LUMPS) = C*XB(2)
      YA(LUMPS) = UL*CA(LUMPS)
      YB(LUMPS) = UL*CB(LUMPS)
      Z = 0.0

C******ICR IS INDEX INDICATING U=0 IN BED

      ICR = 1
      ICR1 = ICR - 1

C******SOLVE FOR U IN BED

      DO 3 I=2,LUMPS
      IM1 = I-1
      Z = Z+DZ
      IF ( IM1-LUM2 ) 35,34,34
35    U(I) = U(1) +BETA*( GI(I)/C - Z )
      GO TO 36
34    U(I) = AR*U(LUM2) + BETA*( (GI(I)-GIL2)/C - Z + L2 )
36    CONTINUE

C******CHECK SIGN CHANGE OF U

      IF ( U(I)*U(IM1) ) 8,9,3
8     ICR = I
      ICR1 = ICR-1

C******CALCULATE INTERIOR INITIAL FLUX ( UO<0, UL>0 )

      ULAST = U(IM1)
      IF ( IM1 .EQ. LUM2 ) ULAST=AR*ULAST
      DZCR = DZ*U(I) / ( U(I)-ULAST )
      DTC = DZCR - DZ
      R1 = U(ICR) / DZCR
      R3A = ( G(I,2)-G(IM1,2) ) / DZ
      R3B = ( G(I,3)-G(IM1,3) ) / DZ
      R2A = G(IM1,2) - R3A*DTC
      R2B = G(IM1,3) - R3B*DTC
      A1A = BETA*R2A*R1 / ( BETA+R1 )
      A1B = BETA*R2B*R1 / ( BETA+R1 )
      A2A = BETA*R3A*R1 / ( BETA+2.*R1 )
      A2B = BETA*R3B*R1 / ( BETA+2.*R1 )
      YA(ICR) = A1A*DZCR + A2A*DZCR*DZCR
      YB(ICR) = A1B*DZCR + A2B*DZCR*DZCR
      YA(ICR1) = A1A*DTC + A2A*DTC*DTC
      YB(ICR1) = A1B*DTC + A2B*DTC*DTC
      CA(ICR) = YA(ICR) / U(ICR)
      CB(ICR) = YB(ICR) / U(ICR)
      CA(ICR1) = YA(ICR1) / ULAST
      CB(ICR1) = YB(ICR1) / ULAST
      GO TO 3
9     ICR = I
      ICR1 = I
      YA(ICR) = 0.0
      YB(ICR) = 0.0
      IF ( UO ) 3,25,3
25    ICR = 1
```

```
      ICR1 = 1
      YA(1) = 0.0
      YB(1) = 0.0
3     CONTINUE
      UL = U(LUMPS)
C------LOGIC FOR FORWARD INTEGRATION
      ISIG = 1
      IF ( UL ) 14,14,15
14    IST = 1
      IF ( UO ) 12,20,21
21    CONTINUE
      IT = ICR-2
      IF ( IT ) 12,12,11
20    ICR = 1
      ICR1 = 1
      GO TO 12
15    CONTINUE
      IST = ICR
      IT = LUMPS-ICR
      IF ( IT ) 12,12,11
11    CONTINUE

C------SOLVE FOR CA

      DO 10 J=1,IT
      I = IST + ISIG*J
      IM1 = I-ISIG
      IF ( IM1-LUM2 ) 33,30,33
30    IF ( ISIG ) 31,33,32
31    YA(IM1) = YA(IM1) / AR
      YB(IM1) = YB(IM1) / AR
      U(IM1) = U(IM1) / AR
      GO TO 33
32    U(IM1) = AR*U(IM1)
      YA(IM1) = AR*YA(IM1)
      YB(IM1) = AR*YB(IM1)
33    CONTINUE
      RO = U(IM1)
      IF ( I .EQ. LUM2 .AND. ISIG .EQ. -1 ) U(I)=AR*U(I)
      R1 = ( U(I)-RO ) / DZ
      R2A = G(IM1,2)
      R2B = G(IM1,3)
      R3A = ( G(I,2)-R2A ) / DZ
      R3B = ( G(I,3)-R2B ) / DZ
      IF ( R1 ) 6,7,6
7     AOA = ( R2A-R3A*RO/BETA ) * RO
      AOB = ( R2B-R3B*RO/BETA ) * RO
      A1A = R3A*RO
      A1B = R3B*RO
      BR = BETA / RO
      YA(I) = ( YA(IM1)-AOA )*EXP(-BR*DZ) + AOA + A1A*DZ
      YB(I) = ( YB(IM1)-AOB )*EXP(-BR*DZ) + AOB + A1B*DZ
      GO TO 10
6     A2A = BETA*R3A*R1 / ( BETA+2.*R1 )
      A2B = BETA*R3B*R1 / ( BETA+2.*R1 )
      A1A = ( BETA*(R3A*RO+R2A*R1 ) - 2.*A2A*RO ) / ( BETA+R1 )
      A1B = ( BETA*(R3B*RO+R2B*R1 ) - 2.*A2B*RO ) / ( BETA+R1 )
      AOA = ( BETA*RO-R2A-A1A*RO ) / BETA
      AOB = ( BETA*RO-R2B-A1B*RO ) / BETA
      IF ( RO ) 26,27,26
27    YA(I) = A1A*DZ + A2A*DZ*DZ
      YB(I) = A1B*DZ + A2B*DZ*DZ
      GO TO 28
```

```
26      CONTINUE
        BR = BETA/R1
        YA(I) = ( YA(IM1)-AOA )*( 1.+R1*DZ/RO )**(-BR) + AOA + A1A*DZ
              + A2A*DZ*DZ
        YB(I) = ( YB(IM1)-AOB )*( 1.+R1*DZ/RO )**(-BR) + AOB + A1B*DZ
              + A2B*DZ*DZ
28      CONTINUE
        IF ( U(I) ) 18,19,18
19      CA(I) = C*G(I,2) / G(I,1)
        CB(I) = C*G(I,3) / G(I,1)
        GO TO 10
18      CA(I) = YA(I) / U(I)
        CB(I) = YB(I) / U(I)
10      CONTINUE
        IF ( ICR .EQ. 1 ) GO TO 13
        IF ( DZ ) 13,13,12
12      CONTINUE

C------LOGIC FOR BACKWARD INTEGRATION

        DZ = -DZ
        ISIG = -1
        IF ( UL ) 16,16,17
16      IST = LUMPS
        IT = LUMPS - ICR
        IF ( ICR .EQ. ICR1 )  IT=IT-1
        IF ( IT ) 13,13,11
17      CONTINUE
        IST = ICR1
        IT = ICR1-1
        IF ( IT ) 13,13,11
13      CONTINUE
        XA(1) = CA(1) / C
        XA(2) = CA(LUMPS) / C
        XB(1) = CB(1) / C
        XB(2) = CB(LUMPS) / C
        DZ = FLOAT(ISIG) * DZ
        XAO = XA(1)
        XAL = XA(2)
        XBO = XB(1)
        XBL = XB(2)
        RETURN
        END
        SUBROUTINE CAPZ ( N, G, I )
        REAL KA,KB,KC,N,NA,NB,NC
        DIMENSION N(300,3),G(300,3)
        COMMON/CAP/KA,KB,KC,BB,BC,D
        NA = N(I,2)
        NB = N(I,3)
        NC = N(I,1) - NA - NB
        G(I,2) = KA*NA
        G(I,3) = KB*NB / (1.-NB/BB )
        GC = KC*NC / (1.-NC/BC )
        G(I,1) = G(I,2) + G(I,3) + GC
        RETURN
        END
        SUBROUTINE VALVE ( A, PU, PD, W )
C--------CALCULATES VALVE MASS FLOWRATE  KG/SEC
        REAL KI
        DATA KI, C1, C2, CK / .714, .00906, .00234, .286 /
        PUL = PU
        PDL = PD
        PR = PDL / PUL
```

```
      SIG = 1.
      IF ( 1.-PR ) 4,3,2
3     W = 0.
      RETURN
4     PR = 1. / PR
      PUL = PDL
      SIG = -1.
2     CONTINUE
      IF ( PR .LT. .528 ) GO TO 1
      PRR = 1. - PR**CK
      W = SIG*A*C1*PUL*(PR**KI)*SQRT(PRR)
      RETURN
1     W = SIG*A*C2*PUL
      RETURN
      END
      SUBROUTINE VAQ ( CG1, CG2 )
C------ITERATION AND FALSE POSITION TO SOLVE VALVE/BED EQUATIONS
      EXTERNAL F1,F2
      COMMON/BEDV/LUMPS,DZ,L,D1,D2,MA,MB,MC,RT,AI,AO,AMWO1,AMWL1,AMWO2
     1 ,AMWL2,WBR
      COMMON/VBEQ/W1,W2,W3,UO1,UL1,UO2,UL2,PN1,PN2,VA1,VA2,VA3
     1 ,BGI1L,BGI1L2,BGI2L,BGI2L2,BL21,BL22,P1,P2
      C1 = CG1
      C2 = CG2
      P1 = C1*RT
      PO1 = P1
      J = 1
10    FC2 = F2 ( C2 )
1     CN2 = C2*( 1.-FC2 )
      IF ( ABS(CN2-C2)/C2 .LE. 1.E-6 ) GO TO 2
      FCN2 = F2 ( CN2 )
      IF ( FC2*FCN2 ) 4,3,3
3     C2 = CN2
      FC2 = FCN2
      GO TO 1
4     C2 = FP ( F2, FCN2, CN2, FC2, C2 )
2     CONTINUE
      FC1 = F1 ( C1 )
5     CN1 = C1*( 1.-FC1 )
      IF ( ABS(CN1-C1)/C1 .LE. 1.E-6 ) GO TO 6
      FCN1 = F1 ( CN1 )
      IF ( FC1*FCN1 ) 8,7,7
7     C1 = CN1
      FC1 = FCN1
      GO TO 5
8     C1 = FP (F1, FCN1, CN1, FC1, C1 )
6     IF ( ABS(P1-PO1)/P1 .LE. 1.E-6 ) GO TO 9
      J = J+1
      PO1 = P1
      IF ( J .LE. 20 ) GO TO 10
      PRINT 100
100   FORMAT ( 1X, '*FAILED TO CONVERGE*' )
9     CG1 = C1
      CG2 = C2
      RETURN
      END
      FUNCTION F1 ( CG1 )
C------BED 1 VALVE/BED EQUATION
      COMMON/BEDV/LUMPS,DZ,L,D1,D2,MA,MB,MC,RT,AI,AO,AMWO1,AMWL1,AMWO2
     1 ,AMWL2,WBR
      COMMON/VBEQ/W1,W2,W3,UO1,UL1,UO2,UL2,PN1,PN2,VA1,VA2,VA3
     1 ,BGI1L,BGI1L2,BGI2L,BGI2L2,BL21,BL22,P1,P2
      C1 = CG1
```

```
      P1 = C1*RT
      CALL VALVE ( VA1, PN1, P1, W1 )
      CALL VALVE ( VA2, P1, P2, W2 )
      IF ( P1-P2 ) 1,2,3
1     WBR1 = 0.
      GO TO 4
2     WBR1 = .5*WBR
      GO TO 4
3     WBR1 = WBR
4     CONTINUE
      UO1 = W1 / ( C1*AMWO1*AI )
      UL1 = ( W2+WBR1 ) / ( C1*AMWL1*A0 )
      AR = AI / A0
      F1 = 1. - ( AR*BGI1L2+BGI1L ) / ( (UL1+BL21-AR*(UO1-BL21))*C1 )
      RETURN
      END
      FUNCTION F2 ( CG2 )
C-----BED 2 VALVE/BED EQUATION
      COMMON/BEDV/LUMPS,DZ,L,D1,D2,MA,MB,MC,RT,AI,A0,AMWO1,AMWL1,AMWO2
     1 ,AMWL2,WBR
      COMMON/VBEQ/W1,W2,W3,UO1,UL1,UO2,UL2,PN1,PN2,VA1,VA2,VA3
     1 ,BGI1L,BGI1L2,BGI2L,BGI2L2,BL21,BL22,P1,P2
      C2 = CG2
      P2 = C2*RT
      CALL VALVE ( VA2, P1, P2, W2 )
      CALL VALVE ( VA3, P2, PN2, W3 )
      IF ( P2-P1 ) 1,2,3
1     WBR2 = 0.
      GO TO 4
2     WBR2 = .5*WBR
      GO TO 4
3     WBR2 = WBR
4     CONTINUE
      UO2 = ( W2-WBR2 ) / ( C2*AMWO2*A0 )
      UL2 = W3 / ( C2*AMWL2*AI )
      AR = A0 / AI
      F2 = 1. - ( AR*BGI2L2+BGI2L ) / ( (UL2+BL22-AR*(UO2-BL22))*C2 )
      RETURN
      END
      FUNCTION FP ( F, FXR, XR, FXL, XL )
C-----USES FALSE POSITION TO FIND ZERO
      DATA EPS / 1.E-6 /
      I = 0
3     X = XL -FXL*(XL-XR) / ( FXL-FXR )
      FX = F ( X )
      I = I+1
      IF ( ABS(FX) .LT. EPS ) GOTO 4
      IF ( I .GT. 50 ) GOTO 4
      IF ( FX*FXL ) 1,1,2
1     XR = X
      FXL = FXL*FXR / ( FXR+FX )
      FXR = FX
      GOTO 3
2     XL = X
      FXR = FXR*FXL / ( FXL+FX )
      FXL = FX
      GOTO 3
4     FP = X
      RETURN
      END

      subroutine zetaplt(nplot,ncvs,npts,range,x,y,ny,xaxis,nxaxis,
     .                   yaxis,nyaxis)
```

```
      dimension x(300),y(302,11),c(6),range(4),xaxis(10),yaxis(10)
      data axlenx,axleny / 7.0, 5.0 /
      call plots ( 0, 0, nplot )
      call plot ( 0.5, 0.5, -3 )
      call factor ( 0.65 )
      if ( (range(1).eq.0.) .and. (range(2).eq.0.) ) then
        call scale ( x, axlenx, npts, 1 )
       else
        x(npts+1) = range(1)
        x(npts+2) = (range(2)-range(1))/axlenx
      end if
      if ( (range(3).eq.0.) .and. (range(4).eq.0.) ) then
        call scale ( y, axleny, npts, 1 )
       else
        y(npts+1,1) = range(3)
        y(npts+2,1) = (range(4)-range(3))/axleny
      end if
      if ( ncvs .ge. 2 ) then
        do n = 2,ncvs
           y(npts+1,n) = y(npts+1,1)
           y(npts+2,n) = y(npts+2,1)
        end do
      end if
      call axis ( 0.0, 0.0, xaxis, -nxaxis, axlenx, 0.0,
     .            x(npts+1), x(npts+2) )
      call axis ( 0.0, 0.0, yaxis, nyaxis, axleny, 90.0,
     .            y(npts+1,1), y(npts+2,1) )
      do m = 1,ncvs
         call line ( x, y(1,m), npts, 1, 0, 2 )
      end do
      call plot ( 0.0, 0.0, 999 )
      return
      end

      SUBROUTINE SCALE1 ( Y, YABS )
      COMMON/HILOBED/NBED
      INTEGER YABS
      DY = 1.0 - Y
      DPY = DY * 150.0
      IDY = IFIX ( DPY )
      YABS = IDY + 100
      RETURN
      END
      SUBROUTINE SCALE2 ( Y, YABS )
      COMMON/HILOBED/NBED
      INTEGER YABS
      DY = 1.0 - Y
      DPY = DY * 150.0
      IDY = IFIX ( DPY )
      YABS = IDY + 300
      RETURN
      END
      SUBROUTINE INITIAL1 ( YABS1, YABS2, NBED )
      INTEGER YABS1,YABS2
      IF ( NBED .EQ. 0 ) THEN
      WRITE(9,100) YABS1
      ELSE
      WRITE(9,100) YABS2
      END IF
  100 FORMAT (3X, '  P[275,'I3']' )
      RETURN
      END
      SUBROUTINE INITIAL2 ( YABS1, YABS2, NBED )
```

```
      INTEGER YABS1,YABS2
      IF ( NBED .EQ. 0 ) THEN
      WRITE(9,100) YABS2
      ELSE
      WRITE(9,100) YABS1
      END IF
  100 FORMAT (3X, '  P[575,'I3']' )
      RETURN
      END
      SUBROUTINE DRAWF ( XABS, YABS, NM )
C DRAW VECTORS FROM PREVIOUS CURSOR POSITIONS TO CURRENT POINTS
C FORWARD PLOTTING
      INTEGER XABS, YABS
      WRITE(9,100) NM,XABS,YABS
  100 FORMAT (3X, ' W(R,N'I1')V[]V['I3','I3']' )
      RETURN
      END
      SUBROUTINE DRAWB ( XABS, YABS, NM )
C BACKWARD PLOTTING
      INTEGER XABS,YABS
      WRITE(9,100) NM, XABS, YABS
  100 FORMAT (3X, ' W(R,N'I1')V[]V['I3', 'I3']' )
      RETURN
      END
      SUBROUTINE TXTERASE (NTICKS)
C  ERASE THE VT125 TEXT SCREEN
      LOGICAL ERASE
      DIMENSION ERASE(4)
      DATA ERASE / 27, '[', '2', 'J' /
      IF ( NTICKS.NE.0 ) THEN
      DO I=1,7
      CALL DELAY (NTICKS)
      END DO
      ELSE
      END IF
      TYPE 10, ERASE
   10 FORMAT ( '+', 5A1, $ )
      RETURN
      END
      SUBROUTINE REGISTART
C  SETS VT125 INTO GRAPHICS(REGIS) MODE
      LOGICAL GRAPHIC
      DIMENSION GRAPHIC(3)
      DATA GRAPHIC / 27, 'P', 'p' /
      TYPE 10, GRAPHIC
   10 FORMAT ( '+', 5A1, $ )
      RETURN
      END
      SUBROUTINE PLTERASE
C  ERASES PLOTS
      LOGICAL ERASE
      DIMENSION ERASE(4)
      DATA ERASE / 'S', '(', 'E', ')' /
      TYPE 10, ERASE
   10 FORMAT ( '+', 5A1, $ )
      RETURN
      END
      SUBROUTINE FRAME1
      WRITE(9,110)
  110 FORMAT ( ' P[250,5]T(S2)"OBOGS SIMULATION"' )
      WRITE(9,112)
  112 FORMAT ( ' P[30,30]T(S1)"TIME =          SEC. "' )
      WRITE(9,114)
```

```
114   FORMAT ( ' P[260,95]T"1"P[260,245]T"0"' )
      WRITE(9,116)
116   FORMAT ( ' P[260,295]T"1"P[260,445]T"0"' )
      WRITE(9,120)
120   FORMAT ( ' P[400,80]T"BED #1"' )
      WRITE(9,130)
130   FORMAT ( ' P[400,280]T"BED #2"' )
      WRITE(9,140)
140   FORMAT ( ' P[20,255]T"INLET"' )
      WRITE(9,150)
150   FORMAT ( ' P[5,275]T"PRESSURE"' )
      WRITE(9,160)
160   FORMAT ( ' P[195,112]T"R.V."' )
      WRITE(9,170)
170   FORMAT ( ' P[95,430]T"EXHAUST"' )
      WRITE(9,180)
180   FORMAT ( ' P[115,450]T"GAS"' )
      WRITE(9,190)
190   FORMAT ( ' P[624,268]T"P.O."' )
      WRITE(9,192)
192   FORMAT ( ' P[680,112]T"B.P."' )
      WRITE(9,193)
193   FORMAT ( ' P[720,412]VOOOOOP[730,403]T"0%"' )
      WRITE(9,195)
195   FORMAT ( ' P[720,356]VOOOOOP[730,349]T"20%"' )
      WRITE(9,196)
196   FORMAT ( ' P[720,132]VOOOOOP[730,123]T"100%"' )
      WRITE(9,200)
200   FORMAT ( ' P[275,100]V[575,100]V[575,250]V[275,250]V[275,100]')
      WRITE(9,210)
210   FORMAT ( ' P[275,300]V[575,300]V[575,450]V[275,450]V[275,300]' )
      WRITE(9,300)
300   FORMAT ( ' P[575,162]V[670,162]V[670,132]V[720,132]V[720,412]' )
      WRITE(9,310)
310   FORMAT ( ' V[670,412]V[670,382]V[575,382]P[575,182]' )
      WRITE(9,400)
400   FORMAT ( ' V[600,182]V[600,267]V[605,272]V[600,277]V[600,362]' )
      WRITE(9,410)
410   FORMAT ( ' V[575,362]P[620,182]V[670,182]V[670,362]V[620,362]' )
      WRITE(9,450)
450   FORMAT ( ' V[620,277]V[615,272]V[620,267]V[620,182]')
      WRITE(9,460)
460   FORMAT ( ' P[720,262]V[750,262]V[750,282]V[720,282]' )
      WRITE(9,500)
500   FORMAT ( ' P[275,162]V[240,162]V[240,132]V[180,132]' )
      WRITE(9,510)
510   FORMAT ( ' V[180,162]V[118,162]V[118,262]' )
      WRITE(9,520)
520   FORMAT ( ' P[275,182]V[240,182]P[180,182]V[138,182]V[138,262]
     1    ' )
      WRITE(9,540)
540   FORMAT ( ' P[118,282]V[118,420]V[138,420]V[138,382]' )
      WRITE(9,600)
600   FORMAT ( ' V[180,382]V[180,412]V[240,412]V[240,382]V[275,382]')
      WRITE(9,610)
610   FORMAT ( ' P[180,262]V[80,262]V[80,282]' )
      WRITE(9,700)
700   FORMAT ( ' V[180,282]V[180,362]V[138,362]V[138,282]P[275,362]' )
      WRITE(9,710)
710   FORMAT ( ' P[275,362]V[240,362]V[240,182]P[180,182]V[180,262]' )
      WRITE(9,730)
730   FORMAT ( ' P[630,162]V[630,167]V[640,162]' )
      WRITE(9,735)
```

```
735  FORMAT ( ' P[630,182]V[630,177]V[640,182]' )
     WRITE(9,740)
740  FORMAT ( ' P[630,362]V[630,367]V[640,362]' )
     WRITE(9,750)
750  FORMAT ( ' P[630,382]V[630,377]V[640,382]' )
     RETURN
     END
     SUBROUTINE NEWTIME ( TIME,NW )
     WRITE(9,100) NW,TIME
100  FORMAT ( ' W(N'I1')P[110,30]T(S1)"'F8.2'"' )
     RETURN
     END
     SUBROUTINE FRAME2 ( NBED,NW )
     IF ( NBED .EQ. 1 ) THEN
     WRITE(9,710)NW
710  FORMAT ( ' W(N'I1')P[180,262]V[240,362]' )
     ELSE
     WRITE(9,720)NW
720  FORMAT ( ' W(N'I1')P[180,262]V[240,162]' )
     END IF
     WRITE(9,730)NW
730  FORMAT ( ' W(N'I1')P[630,162]V[630,167]V[640,162]' )
     WRITE(9,735)
735  FORMAT ( ' P[630,182]V[630,177]V[640,182]' )
     WRITE(9,740)NW
740  FORMAT ( ' W(N'I1')P[630,362]V[630,367]V[640,362]' )
     WRITE(9,750)
750  FORMAT ( ' P[630,382]V[630,377]V[640,382]' )
     IF ( NBED .EQ. 1 ) THEN
     WRITE(9,800)NW
800  FORMAT ( ' W(N'I1')P[180,282]V[240,382]' )
     WRITE(9,805)
805  FORMAT ( ' P[240,162]V[180,162]P[240,182]V[180,182]' )
     WRITE(9,810)NW
810  FORMAT ( ' W(N'I1')P[113,272]V[143,272]' )
     WRITE(9,815)
815  FORMAT ( ' P[136,268]V[]V[143,272]V[136,276]' )
     WRITE(9,820)NW
820  FORMAT ( ' W(N'I1')P[590,372]V[625,372]' )
     WRITE(9,825)
825  FORMAT ( ' P[620,369]V[]V[625,372]V[620,375]' )
     WRITE(9,830)NW
830  FORMAT ( ' W(N'I1')P[610,320]V[610,290]' )
     WRITE(9,835)
835  FORMAT ( ' P[607,295]V[]V[610,290]V[613,295]' )
     WRITE(9,840)NW
840  FORMAT ( ' W(N'I1')P[610,172]V[585,172]' )
     WRITE(9,845)
845  FORMAT ( ' P[590,169]V[]V[585,172]V[590,175]' )
     WRITE(9,850)NW
850  FORMAT ( ' W(N'I1')P[128,352]V[128,392]' )
     WRITE(9,855)
855  FORMAT ( ' P[124,385]V[]V[128,392]V[132,385]' )
     WRITE(9,860)NW
860  FORMAT ( ' W(N'I1')P[638,172]C[632,166]' )
     WRITE(9,865)
865  FORMAT ( ' P[645,172]V11117777111177771111000' )
     WRITE(9,870)NW
870  FORMAT ( ' W(N'I1')P[650,372]C[643,372]' )
     WRITE(9,880)
880  FORMAT ( ' P[657,372]V11117777111100' )
     ELSE
     WRITE(9,900)NW
```

```
900   FORMAT ( ' W(N'I1')P[180,282]V[240,182]' )
      WRITE(9,905)
905   FORMAT ( ' P[180,362]V[240,362]P[180,382]V[240,382]' )
      WRITE(9,910)NW
910   FORMAT ( ' W(N'I1')P[113,272]V[143,272]' )
      WRITE(9,915)
915   FORMAT ( ' P[138,268]V[]V[143,272]V[136,276]' )
      WRITE(9,920)NW
920   FORMAT ( ' W(N'I1')P[590,172]V[615,172]' )
      WRITE(9,925)
925   FORMAT ( ' P[608,168]V[]V[615,172]V[608,176]' )
      WRITE(9,930)NW
930   FORMAT ( ' W(N'I1')P[610,222]V[610,252]' )
      WRITE(9,935)
935   FORMAT ( ' P[606,245]V[]V[610,252]V[614,245]' )
      WRITE(9,940)NW
940   FORMAT ( ' W(N'I1')P[610,372]V[580,372]' )
      WRITE(9,945)
945   FORMAT ( ' P[587,368]V[]V[580,372]V[587,376]' )
      WRITE(9,950)NW
950   FORMAT ( ' W(N'I1')P[159,372]V[128,372]V[128,390]' )
      WRITE(9,955)
955   FORMAT ( ' P[124,381]V[]V[128,390]V[132,381]' )
      WRITE(9,960)NW
960   FORMAT ( ' W(N'I1')P[650,172]C[643,172]' )
      WRITE(9,965)
965   FORMAT ( ' P[657,172]V11117777111100' )
      WRITE(9,970)NW
970   FORMAT ( ' W(N'I1')P[638,372]C[633,366]' )
      WRITE(9,980)
980   FORMAT ( ' P[645,372]V11117777111177770000' )
      END IF
      RETURN
      END
      SUBROUTINE REGISOUT
C  SETS VT125 BACK INTO TEXT MODE
      LOGICAL ALPHA
      DIMENSION ALPHA(2)
      DATA ALPHA / 27, '\' /
      TYPE 10, ALPHA
 10   FORMAT ( '+', 5A1, $ )
      RETURN
      END
      SUBROUTINE DELAY ( NTICKS )
      TYPE 100,NTICKS
 100  FORMAT ( '+S(T<'I3'>)', $ )
      RETURN
      END
      SUBROUTINE SCALEM ( Y,IYABS,VV,RANGE )
      DY = (( VV - Y ) * 300.0 ) / RANGE
      IDY = IFIX ( DY )
      IYABS = IDY + 90
      RETURN
      END
      SUBROUTINE STIME ( TIME,IXABS,RANGE,AXLEN,SHIFT )
      INTEGER SHIFT
      DX = ( TIME * AXLEN ) / RANGE
      IDX = IFIX ( DX )
      IXABS = IDX + SHIFT
      RETURN
      END
      SUBROUTINE FRAME
      COMMON/SYST/DT,PSUP,POUT,TF,WBRL,VS,V0,IIN,TCYC
```

```
      COMMON/NPRINT/NOUT1
      DIMENSION T(10)
      DELT = TF/10.0
      DO I=1,10
      T(I)=I*DELT
      END DO
      TYPE 10
10    FORMAT ( '!P[130,390]V[+60,]V[,+5]' , $ )
      DO I=1,9
      TYPE 12
12    FORMAT ( '!P[,-5]V[+60,]V[,+5]' , $ )
      END DO
      TYPE 20
20    FORMAT ( '!P[120,398]T"0.0"' , $ )
      TYPE 21 , T(1),T(2)
21    FORMAT ( '!P[160,398]T"'F5.1'"P[220,398]T"'F5.1'"' , $ )
      TYPE 22 , T(3),T(4)
22    FORMAT ( '!P[280,398]T"'F5.1'"P[340,398]T"'F5.1'"' , $ )
      TYPE 23 , T(5),T(6)
23    FORMAT ( '!P[400,398]T"'F5.1'"P[460,398]T"'F5.1'"' , $ )
      TYPE 24 , T(7),T(8)
24    FORMAT ( '!P[520,398]T"'F5.1'"P[580,398]T"'F5.1'"' , $ )
      TYPE 25 , T(9),T(10)
25    FORMAT ( '!P[640,398]T"'F5.1'"P[700,398]T"'F5.1'"' , $ )
      TYPE 30
30    FORMAT ( '!P[350,430]T"TIME (SEC.)"', $ )
      TYPE 40
40    FORMAT ( '!P[130,390]V[,-30]V[-5,]' , $ )
      DO I=1,9
      TYPE 50
50    FORMAT ( '!P[+5,]V[,-30]V[-5,]' , $ )
      END DO
      TYPE 51
51    FORMAT ( '!P[70,180]T[+0,+16]"OUTPUT"' , $ )
      TYPE 53 , PSUP
53    FORMAT ( '!P[530,34]T[+9,+0]"PSUP = "T"'F5.2'"T" PSIA"',$)
      TYPE 54 , POUT
54    FORMAT ( '!P[530,52]T"POUT = "T"'F5.2'"T" PSIA"',$)
      TYPE 55 , WBRL
55    FORMAT ( '!P[530,70]T"WBRL = "T"'F5.2'"T" STD LIT/MIN"',$)
      IF(NOUT1.EQ.1)THEN
57    TYPE 60
60    FORMAT ( '!P[290,5]T"* OXYGEN MOLE FRACTION *"' , $ )
      TYPE 62
62    FORMAT ( '!P[95,383]T"0.0"P[95,353]T"0.1"P[95,323]T"0.2"'$)
      TYPE 64
64    FORMAT ( '!P[95,293]T"0.3"P[95,263]T"0.4"P[95,233]T"0.5"',$)
      TYPE 66
66    FORMAT ( '!P[95,203]T"0.6"P[95,173]T"0.7"P[95,143]T"0.8"',$)
      TYPE 68
68    FORMAT ( '!P[95,113]T"0.9"P[95,83]T"1.0"',$)
      ELSE IF(NOUT1.EQ.2) THEN
      TYPE 80
80    FORMAT ( '!P[290,5]T"* INLET MASS FLOWRATE (KG/SEC) *"',$)
      TYPE 82
82    FORMAT ( '!P[85,383]T"-.02"P[85,353]T"-.01"P[95,323]T"0.0"',$)
      TYPE 84
84    FORMAT ( '!P[95,293]T".01"P[95,263]T".02"P[95,233]T".03"',$)
      TYPE 86
86    FORMAT ( '!P[95,203]T".04"P[95,173]T".05"P[95,143]T".06"',$)
      TYPE 88
88    FORMAT ( '!P[95,113]T".07"P[95,83]T".08"',$)
      ELSE IF(NOUT1.EQ.3) THEN
```

```
      TYPE 90
90    FORMAT ( '!P[290,5]T"- OUTLET MASS FLOWRATE (KG/SEC) -"',$)
      TYPE 82
      TYPE 84
      TYPE 86
      TYPE 88
      ELSE
      GO TO 57
      END IF
      RETURN
      END
```

APPENDIX C.

These are the corrected equations for the van der Vlist paper (8). These are for the binary mixture at total pressure (constant) P. The six unknowns (A's and B's) are found in terms of known quantities (G's and W's). See the paper for the complete details. Essentially, six equations in six unknowns are solved. The equations are:

$$G_0 = P * A_1 \tag{1}$$

$$G_1 = 2 * P * A_2 - A_1 + B_1 + 2 * P * B_2 + 3 * P^2 * B_3 \tag{2}$$

$$G_2 = 3 * P * A_3 - 2 * A_2 - 2 * B_2 - 3 * 2 * P * B_3 \tag{3}$$

$$G_3 = - 3 * A_3 + 3 * B_3 \tag{4}$$

$$W_1 = P * A_1 + P^2 * A_2 + P^3 * A_3 \tag{5}$$

$$W_2 = P * B_1 + P^2 * B_2 + P^3 * B_3 \tag{6}$$

These equations give the following solution:

$$A_1 = \frac{G_0}{P}$$

$$A_2 = \frac{- 6 * W_2 - 3 * W_1 + 2 * G_3 * P^3 + 3 * G_2 * P^2 + 6 * G_1 * P + 9 * G_0}{3 * P^2}$$

$$A_3 = - \frac{- 6 * W_2 - 6 * W_1 + 2 * G_3 * P^3 + 3 * G_2 * P^2 + 6 * G_1 * P + 12 * G_0}{3 * P^3}$$

$$B_1 = \frac{G_3 * P^3 + G_2 * P^2 + G_1 * P + G_0}{P}$$

$$B_2 = - \frac{3 * W_2 + 6 * W_1 + 2 * G_3 * P^3 - 3 * G_1 * P - 9 * G_0}{3 * P^2}$$

$$B_3 = - \frac{- 6 * W_2 - 6 * W_1 + G_3 * P^3 + 3 * G_2 * P^2 + 6 * G_1 * P + 12 * G_0}{3 * P^3}$$

DISTRIBUTION LIST

1 copy	Commander US Army Medical Research and Development Command ATTN: SGRD-RMI-S Fort Detrick Frederick, MD 21701-5012
1 copy	Commander US Army Aeromedical Research Laboratory ATTN: SGRD-UAB/Dr. Kent Kimball Fort Rucker, AL 36362-5000
12 copies	Administrator Defense Technical Information Center ATTN: DTIC-DDA Cameron Station Alexandria, VA 22314
1 copy	Commandant US Army Academy of Health Sciences ATTN: AHS-CDM Fort Sam Houston, TX 78234
1 copy	Dean, School of Medicine Uniformed Services University of the Health Services 4301 Jones Bridge Road Bethesda, MD 20014